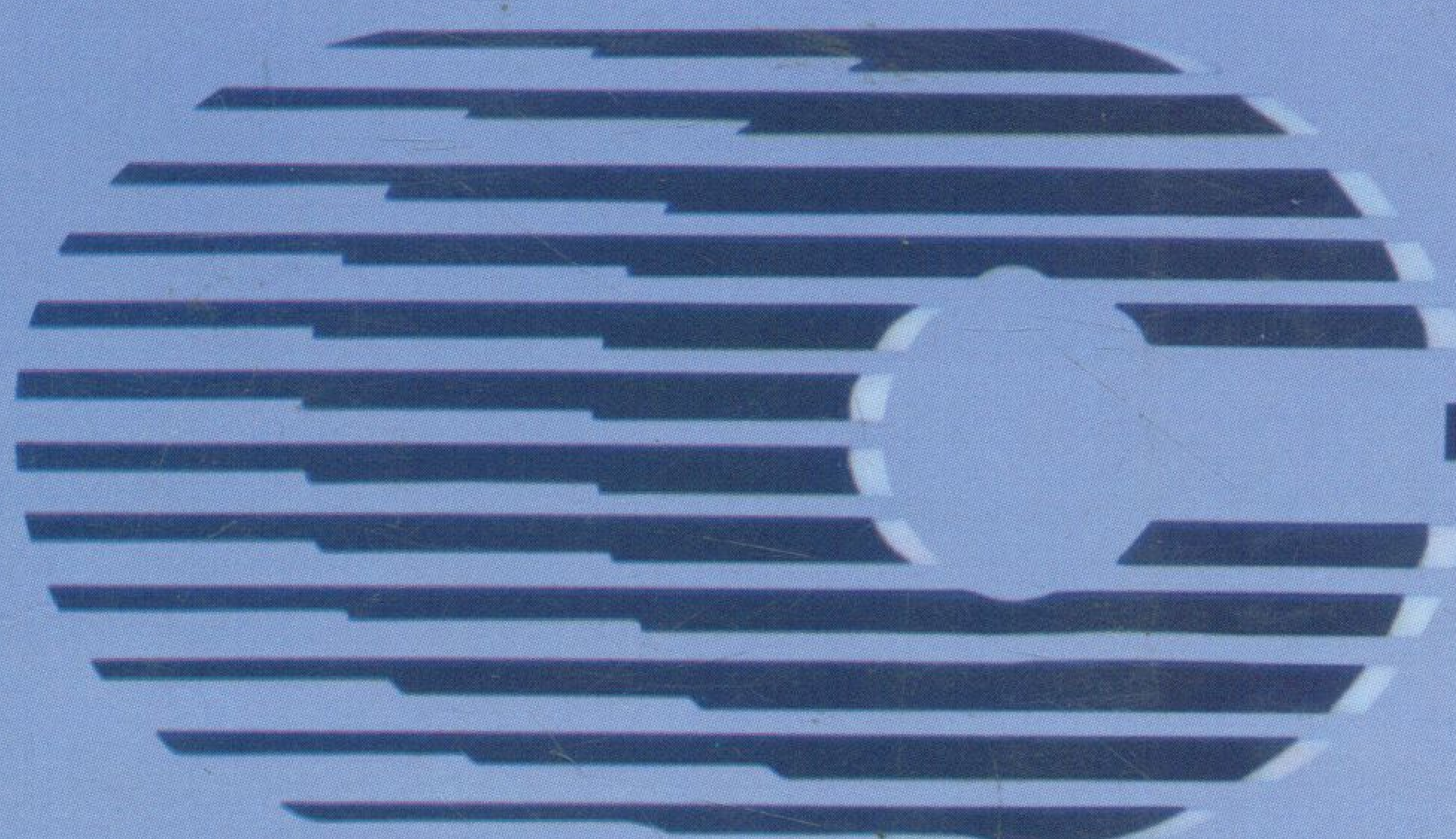
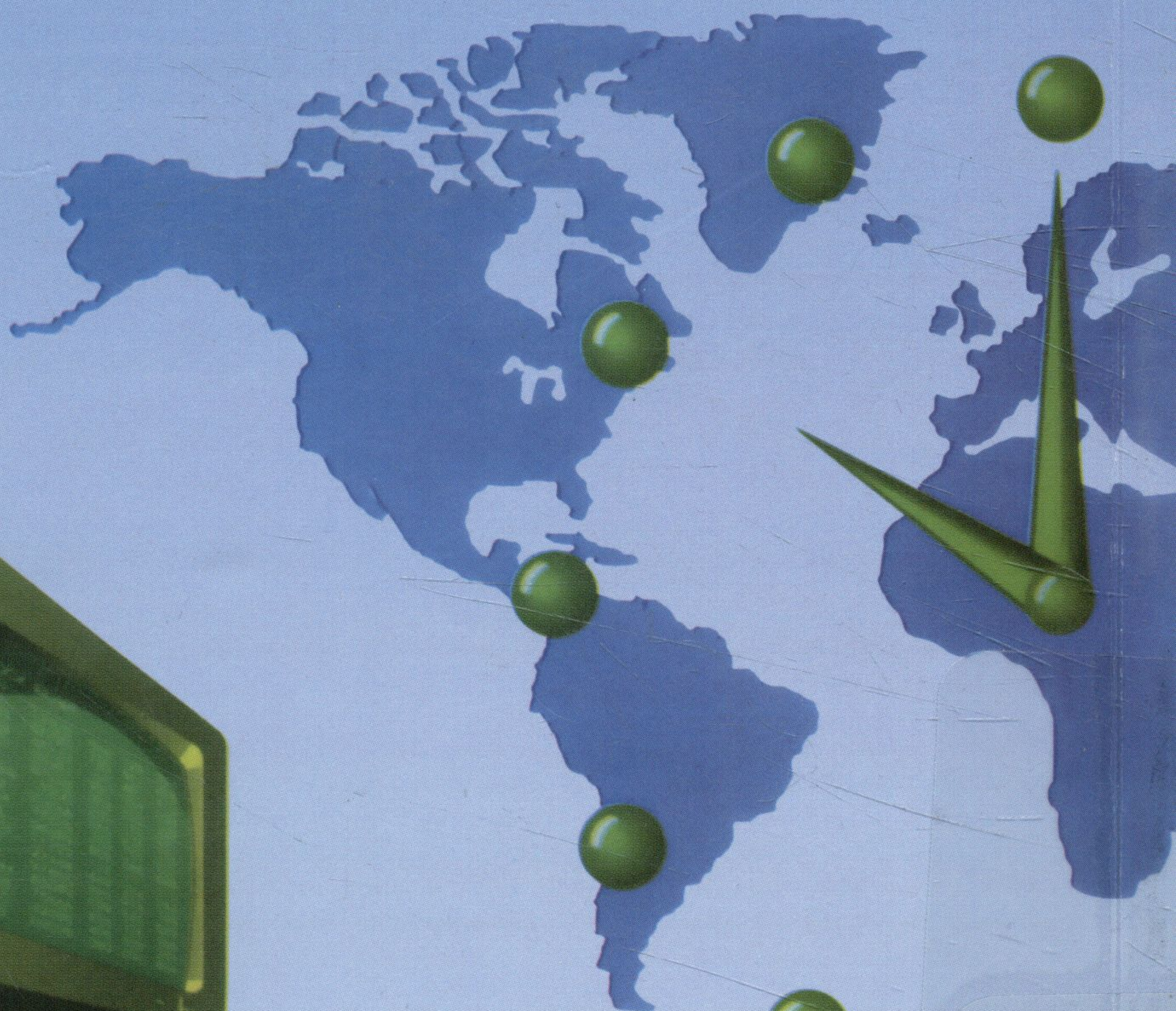
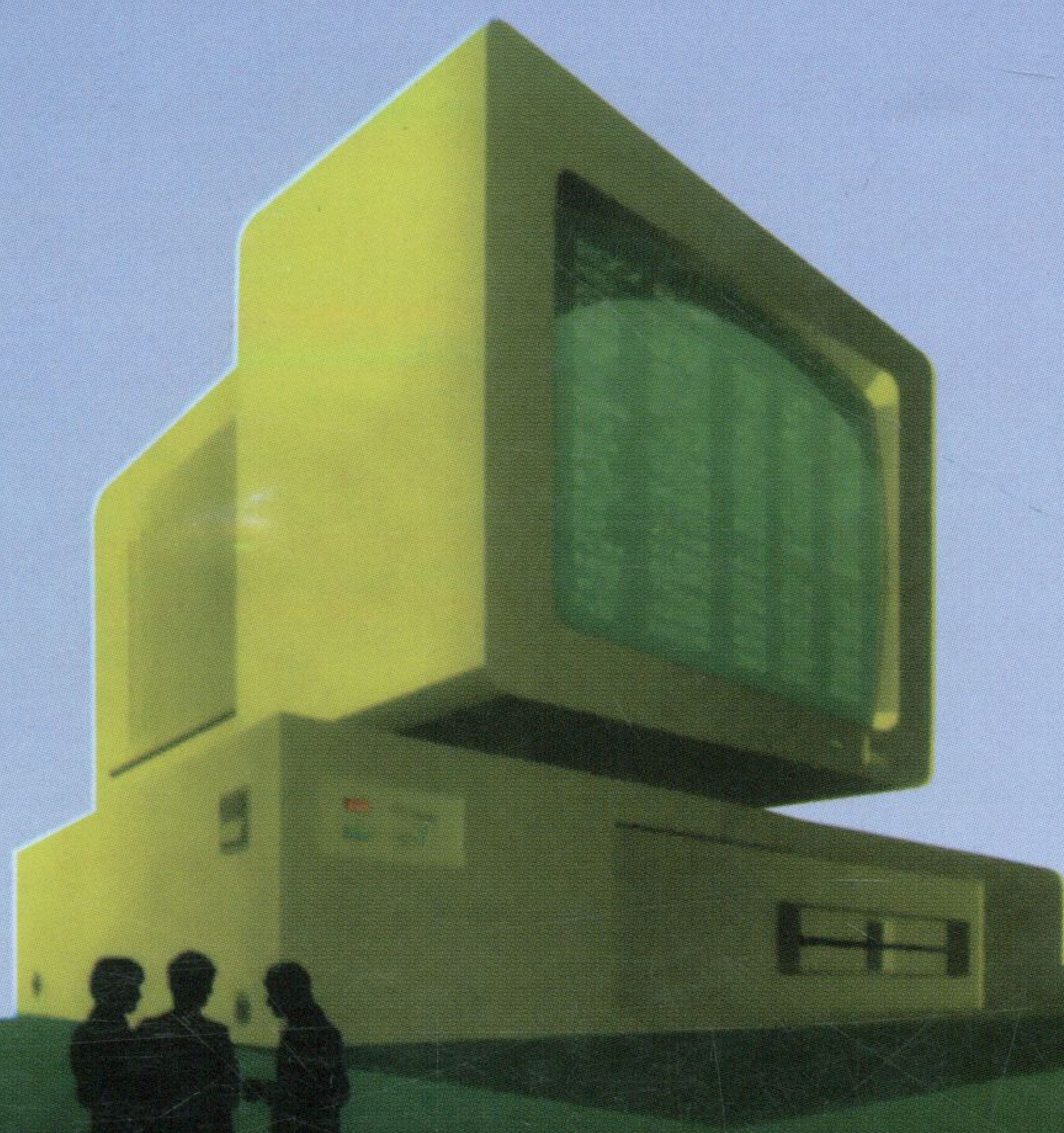
أساليب البرمجة بلغة









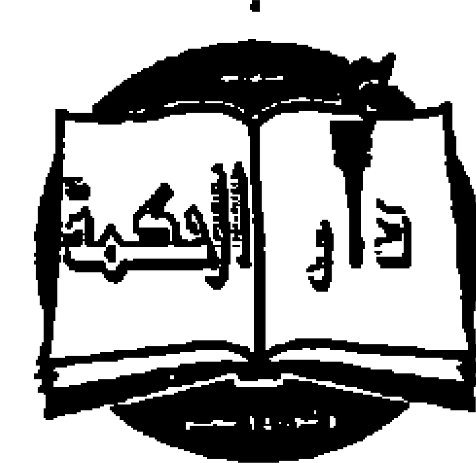
بشير علي القائد

أساليب البرمجة بلغة
C++

بشير علي القائد
أستاذ مشارك - قسم الحاسب الآلي
كلية العلوم - جامعة طرابلس
ليبيا

الطبعة الرابعة 2014

رقــــم الايـــداع : 2013-564
الرقم الدولي (ردمك): ISBN 978-9959-808-51-6
الوكالة الليبية للترقيم الدولي الموحد للكتاب.
دار الكتاب الوطني – بنغازي – ليبيا
هاتف: 9090509 – 9096379 – 9097074.
بريد مصور : 9097073



دار الحكمة للطباعة و النشر و التوزيع

هاتف : 0213606610 0213606571

نقال : 00218918127948

E-mail: daralhikmabookshop@yahoo.com

طرابلس – ليبيا

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

﴿ رَبِّ أَوۡزِعۡنِيٓ أَنۡ أَشۡكُرَ نِعۡمَتَكَ ٱلَّتِيٓ أَنۡعَمۡتَ
عَلَيَّ وَعَلَىٰ وَٰلِدَيَّ وَأَنۡ أَعۡمَلَ صَٰلِحٗا تَرۡضَىٰهُ وَأَصۡلِحۡ لِي فِي
ذُرِّيَّتِيٓۖ إِنِّي تُبۡتُ إِلَيۡكَ وَإِنِّي مِنَ ٱلۡمُسۡلِمِينَ ﴾

صدق الله العظيم

سورة الأحقاف من الآية 14

مقدمة

بسم الله الرحمن الرحيم

الحمد لله رب العالمين الذي علم الإنسان ما لم يكن يعلم، وكان فضله علينا عظيماً، ومن مظاهر فضله على الإنسان أن زوده بهذه الطاقة المعنوية الرهيبة التي هي العقل والتي استطاع بواسطتها اكتشاف هذه القوة المادية الهائلة وهو الحاسب الآلي الذي أصبح الرمز الأول لهذه الحضارة ، فقد أصبح الإنسان المعاصر بفضل هذا الاكتشاف الهائل قادراً على اختصار المسافات والأزمنة وجعلها طوع إرادته .

وبعد

فاني وبحمد الله وتوفيقه ، أقدم هذا الكتاب الذي يتناول واحدة من أهم وأكثر لغات الحاسب الآلي الرائدة في مجال البرمجة في عالمنا المعاصر وهي لغة سي++ في طبعته الثانية ، حيث قمنا بمراجعة محتوياته وإدخال بعض التعديلات والتنقيحات عليه والتي لم تكن موجودة في الطبعة الأولى .

واني إذ أتقدم بهذا الجهد المتواضع الذي التزمت فيه بتقديم شروحات لعدد كثير من الأمثلة المحلولة عمليا على جهاز الحاسب الشخصي مع مجموعة من التمرينات التي تساعد على إثراء مادته لآمل أن يسد هذا الكتاب فراغاً

في مكتبتنا العربية في مجال حل المسائل المختلفة وان يجعل منه كتاباً منهجياً لطلابنا في الجامعات والمعاهد ومساعدة القارئ في تعلم هذه اللغة بنفسه حتي يمكنه التعامل مع جهاز الحاسب الذي أصبح من ضروريات حياتنا اليومية.

والله وحده نسأل أن يجعل هذا الجهد المتواضع خالصاً لوجه تعالي وأن يجد الذين يقرأون هذا الكتاب لأنفسهم فيه نفعا وأن يجزي من أسهم في نشره وأعان عليه خير الجزاء وأن يغفر لنا أي خطأ أو تقصير

{ رَبَّنَا لاَ تُؤَاخِذْنَا إِن نَّسِينَا أَوْ أَخْطَأْنَا } .

بشير علي القائد

تاجوراء 2009

المحتويات

الفصل الاول : اساسيات البرمجة بلغة سي++

الفصل الثاني: قنوات ادخال واخراج البيانات

الفصل الثالث: الجمل والمؤثرات

الفصل الرابع: جمل الاختيارات

الفصل الخامس: جمل التكرار والتفرعات

الفصل السادس: دوال معالجة الحرفيات

الفصل السابع: الدوال

الفصل الثامن: مؤثرات التعامل مع البت

الفصل التاسع: المصفوفات

الفصل العاشر: المؤشرات

الفصل الحادي عشر: تراكيب البيانات

الفصل الثاني عشر: معالجة الملفات

الفصل الثالث عشر: الفصائل

الفصل الرابع عشر: الرسومات

الملاحـــق

الفصل الاول: اساسيات البرمجة بلغة سي++

يعتبر الحاسب الآلي من المميزات الكبرى في هذا العصر فهو يكتب ويحسب ويعالج البيانات ويخزن المعلومات ويسترجعها وينفذ العمليات بسرعة فائقة جدا ، وعليه فان هذا الفصل يهدف الى شرح كيفية الدخول الى ساسيات لغة عنوان هذا الكتاب ألا وهي أساليب البرمجة بلغة سي++ .

1.1) اساسيات لغة سي++ (C++ Fundamentals)

يجدر بنا تقديم المباديء الاساسية التي تعتمد عليها لغة سي++ كما هو الحال في اللغات الاخرى المتداولة بين البشر ومنها :-

1) الارقام (Digits) وهي 9,8,7,6,5,4,3,2,1,0
2) الحروف الهجائية (Letters) Z,Y,X,...,C,B,A و z,y,x,...,c,b,a.
3) الرموز الخاصة (Special Characters) وهي غير ما ذكر اعلاه.

2.1) الرموز Characters

الرمز في هذه اللغة هو عبارة عن حرف أو رقم أو رمز خاص موضوع بين علامتي التنصيص الفردية (') حيث يمكن تخزينه ومعالجته بجهاز الحاسب، ويتم تمثيل الرموز في ذاكرة الحاسب على هيئة ارقام ثنائية عن طريق عدد من الشفرات المتقدمة واشهرها الشفرة الامريكية القياسية لتبادل المعلومات (ASCII) وهي اختصار للعبارة:

(American Standard Code for Information Interchange)

انظر الى جدول رموز الشفرة بالملحق (2) مع ما يقابلها من اعداد في الانظمة العددية وهي العشري والثنائي والثماني والستة عشرى، مع الاخذ في الاعتبار ان ألـ 32 رمزا الاولى هي رموز تحكم ولا يمكن طباعتها.

مثال 1-1) الامثلة التالية تمثل بعض أنواع الرموز

'a' 'A' '5' '&'

اضافة الى الحرف توجد السلسلة String وهي التي تتكون من مجموعة من الرموز سواء كانت حروفا أو ارقاما أو رموزا خاصة ، بشرط أن تكون هذه الرموز موضوعة بين علامتي التنصيص المزدوجة (") .

مثال 1-2) فيما يلي بعض الامثلة على السلاسل المقبولة :-

" go to ROOM # 45 " " EMPLOYEE NAME "

" My boy said to me, \' I have a cold today.\ "

"ما هو عنوانك ؟ " " ادخل الاختيار المناسب "

3.1) الاعداد Numbers

العدد هو ذلك المقدار الثابت الذي لا تتغير قيمته ويتكون من مجموعة من الأرقام (Digits) ويكون ذا حد ادني وحد اقصى ، ويعتمد ذلك على نوع الجهاز المستعمل ، وتنقسم الاعداد الى :-

أولا) الاعداد الصحيحة Integer Numbers

العدد الصحيح أيا كان، موجبا أو سالبا، أو حتي صفرا يشترط فيه الا يحتوي على نقطة عشرية أو أسية وألا يكون فيه أي رمز خاص أو حرف هجائي، مع وضع اشارة (-) على يسار العدد لبيان انه سالب.

مثال 1-3) الأعداد التالية تعتبر أعدادا مقبولة:-

123456789 0 -1111

في حين الأعداد التالية تعتبرغير مقبولة ، للأسباب المذكورة أمام كل منها

15.69 لوجود نقطة عشرية (.)

7000,000 لوجود رمز خاص (,)

100$ لا يعتبر رقما لوجود علامة الدولار ($)

وعموما فالأعداد الصحيحة يمكن تصنيفها حسب نوع الحاسب كالآتي:-

1) العدد الصحيح (integer) حيث يخصص له 16 بت (2bytes) أو 32 بت ويعتمد ذلك على نوع جهاز الحاسب.
2) العدد الصحيح القصير (short integer) يخصص له عادة 16 بت ومداه من 32768- الى 32767.
3) العدد الصحيح الطويل (long integer) ويخصص له عادة 32 بت ومداه هو 2147483648- الى 2147483647.

فمثلاً عندما تكون القيمة من 32768- إلى 32767 نستعمل (short integer) فيحجز المترجم مساحة اقصر من (integer)، في حين نستعمل (long integer) عندما نريد حجز مساحة اكبر من (integer) لتخزين عدد صحيح.

هناك أعداد صحيحة بدون اشارة (Unsigned Numbers) وهي الأعداد الصحيحة القصيرة الموجبة (unsigned short integer) التي يخصص لها 16 بت ومداها من صفر الى 65535، ايضا الاعداد الصحيحة الطويلة الموجبة (unsigned long integer) وعادة يكون لها المدي من الصفر الى 4294967295، وفيما يلي أنواع البيانات التي تستخدم مع الاعداد الصحيحة:-

unsigned	long	short	int
unsigned	short	unsigned	long

يجب أن يؤخذ في الاعتبار أنه في حالة كون المدخلات من النوع الصحيح فان لغة سي++ سوف تتعامل معها على أنها أعداد في النظام العشرى الذي يعرف بالأرقام (9,8,7,6,5,4,3,2,1,0) وأساسه الرقم 10 أما اذا بدأ العدد بالرقم (0) ففي هذه الحالة يفسر على أنه في النظام الثماني وأرقامه هي (7,6,5,4,3,2,1,0) وأساسه الرقم 8 أما اذا كان الرقم مبتدئا بـ oX أو بـ ox ففي هذه الحالة يعتبر العدد في النظام الستة عشرى التي يتكون من الأرقام (f,e,d,c,b,a,9,8,7,6,5,4,3,2,1,0) وأساسه الرقم 16 وسوف نتناول هذه الأنظمة بشيء من التفصيل لاحقا ان شاء الله.

مثال 1-4) خذ مثلا، يعتبر العدد 16 من النظام العشري بينما يعتبر العدد 016 في النظام الثماني الذي يكافيء العدد 14 في النظام العشري، وكذلك العدد 0X16 مصنفاً ضمن النظام الستة عشري ويعتبر مكافئا للعدد 22 في النظام العشري.

ثانيا) الأعداد الحقيقية Float Numbers

وهي تلك الاعداد التي بها كسور عشرية ، أي التي تحتوي على نقطة عشرية، ولا يكون فيها أي رمز أو حرف ، وقد يكون العدد موجبا أو سالبا بوضع اشارة (-) قبله ، وتسمح لغة سي++ بثلاثة انواع من هذه الاعداد هي:

- عدد سائب أوعائم float وعادة ما يكون له 16 بت ودقتها 5 اعداد .
- عدد مضاعف double وله 64 بت ودقتها 15 عددا .
- عدد مضاعف طويل long double وله 96 بت أو 128 بت ودقتها 19 عددا.

كما هو الحال في كثير من لغات البرمجة تمكننا لغة سى++ من السماح بتمثيل العدد الحقيقي بأسلوبين :-

1) الاول وهو النقطة الثابتة (fixed point) ويحتوي على:-

- الجزء الصحيح
- الفاصلة العشرية النقطة (.)
- الجزء الكسري ما بعد النقطة

مثال 1-5) فيما يلي بعض الأعداد الثابتة المقبولة:-

0.009   63.0   -11.005   3.45

على حين الأعداد التالية تعتبر غير مقبولة للأسباب المبينة أمام كل منها:

100.50 $ لوجود علامة الدولار

998.77.4 لوجود أكثر من نقطة عشرية .

555 لعدم وجود نقطة عشرية

2) الثاني وهو النقطة السائبة أو العائمة (floating point) ويحتوي :-

- الجزء الصحيح
- الحرف e أو الحرف E
- الجزء الصحيح موجب أو سالب (يمثل القوة الآسية)

مثال 1-6) يمكن تمثيل العدد الحقيقي 14000000000.0 وهو اربعة عشر بليونا بشكل قوة اسية على النحو التالي :-

14E9 أو 14.0e9 أو 0.14E11 أو 14e11. أو 1.4E10 أو 14.E9

في حين ان العدد بالقوة الآسية

3.45E4 يساوي 34500.0

3.45E-4 يساوي 000345.

-3.4567E3 يساوي -3456.7

يأخذ العدد الحقيقي 8 بايت، ويتراوح المدى من 3.4E-38 في الحد الأدني الي المدي 3.4E+38 في الحد الأقصى للأعداد الموجبة، و3.4E-38- في الحد الأدنى و3.4E+38- في الحد الأقصى للاعداد السالبة.

هناك ايضا الأعداد ذات الدقة المضاعفة (double)، وهي ناحية اخرى لتقديم العدد من حيث عدد الخانات، وهي 8 بايت ويتراوح المدى من 1.7E-308 في الحد الأدنى و 1.7E+308- في الحد الأقصى للاعداد الموجبة و 1.7E-308- في الحد الأدنى و 1.7E+308- للاعداد السالبة.

4.1) الكلمات المحجوزة Reserved Words

هناك بعض الكلمات المحجوزة في لغة سي++ تستعمل كأسماء، ولها معنى قياسي، ولا يمكن استعمالها كمتغيرات في البرنامج لانها تسبب ارتباكا للمترجم (Compiler)، وفيما يلي بعض هذه الاسماء:-

asm	auto	break	case	catch	char	const	do
delete	double	else	float	friend	for	goto	if
inline	int	new	long	operator	private	protected	public
register	return	short	size of	static	struct	switch	template
this	typedef	union	unsigned	virtual	void	while	

5.1) المعرفات Identifiers

يقصد بالمعرف ذلك الاسم الذي تحفظ فيه قيم المتغيرات مثل الثابت أو المتغير أو الدالة، ومن شروط المعرف:-

1) أن يتكون المعرف من حرف أو مجموعة حروف أو حروف وارقام وعلامة (_) under score باي ترتيب كان.
2) ينبغي أن يبدأ المعرف بحرف من الجهة اليسرى أو الرمز (_) .

3) يجب أن يكون خاليا من الرموز الخاصة فيما عدا (_) .

4) يسمح باستخدام الحروف الصغيرة والحروف الكبيرة .

ويمكن أن يكون للمعرف الطول المناسب ولكن يجب أن يكون واضحا وذا معني ومدلول.

مثال 1-7) فيما يلي بعض المعرفات التي تعتبر مقبولة :-

EmpNumber   total_amount   area5

installations   REAL   name   SALESTAXRATE

في حين ان المعرفات الآتية غير مقبولة، وذلك للاسباب المبينة قرين كل منها:-

bothyou&me	يوجد به الرمز الخاص &
9 digits	الخانه الاولي رقم وليست حرفا
customer#	ينتهي بالرمز الخاص #
Hourse Rate	لا يسمح باستخدام الفراغ
item-number	لا يسمح باستخدم الرمز الخاص (-)
double	كلمة محجوزة

6.1) المتغيرات Variables

هي اسماء رمزية يخصص لها أماكن تخزين في ذاكرة الحاسب، والتي تتحول قيمتها وتتغير من قيمة الى قيمة اخرى، حيث يمكن الرجوع الى هذه القيم عن طريق هذه الأسماء وذلك اثناء تنفيذ البرنامج.

في لغة سى++ يجب ان يؤخذ في الاعتبار ضرورة الاعلان (Declaration) عن المتغيرات صراحة مسبقا في أي مكان من البرنامج مادام الاعلان سابقا لاستخدام المتغير، والا فلن يعترف بها المترجم (Compiler).

الشكل العام لتعريف المتغير هو:-

Type Variable_1,Variable_2,...;

حيث:

Type تعني نوع المتغير المراد الاعلان عنه ، ويجب ان يكتب النوع بالحروف الصغيرة ويمكن ان يكون من الانواع التالية :-

int or long or unsigned char or char or short or unsigned short or unsigned or unsigned long or float or double

Variable_1, Variable_2,... قائمة بمتغير أو اكثر من متغير يفصل بينها بفواصل ، ويمكن ان تكون حروفا صغيرة أو كبيرة .

ايضا ميزة اخرى موجودة في لغة سى++ وهي الاعلان عن المتغير وتخصيص قيمة له في نفس الوقت والشكل هو الآتي :-

```
type  Variable 1 = InitialValue 1 ;
type  Variable 2 = InitialValue 2 ;

type  Variable n = InitialValue n;
```

والآن وجب علينا ذكر بعض أنواع المتغيرات، واعطاء امثلة عليها

1) المتغيرات الحرفية Character Variables

المتغيرات من هذا النوع يمكن اشهارها أو الاعلان عنها كمتغيرات حرفية عن طريق الكلمة char في البرنامج قبل استخدامها، وعليه فهي تستوعب خانه واحده فقط لكل متغير.

مثال 1-8) الاعلان التالي

```
char a,b;
 a='?';
 b='&';
```

يبين على ان المتغيرين b,a هما من النوع الحرفي ويسمح بتخصيص رمز واحد لكل منهما.

ويمكن تخزين أي رمز في بايت (byte) واحدة أي ثمانية جزئيات (8 bits)، ومن هنا فان قيمتي المتغيرين b,a تكونان مخزنتين في 16 بت كما يلي:-

0011111100100110

حيث البايت الاولى 00111111 لقيمة المتغير a وهي (?) والثانية 00100110 لقيمة b وهي (&).

مثال 1-9) الاعلان

```
char *X= "Tajura - Tripoli";
```

يكافئ

```
char X[15]= " Tajura - Tripoli ";
```

وفي كليهما يعتبر المتغير X من نوع السلسلة الحرفية، حيث يحتوي على 15 حرفاً (14 حرفاً وهو عدد حروف اسم مدينة تاجوراء - طرابلس اضافة الى رمز نهاية هذه السلسلة وهو الرمز \0).

2) المتغيرات الصحيحة Integer Variables

وهي متغيرات يسمح بتخزين العدد الصحيح فيها ، سواء كان العدد صحيحا سالبا أو موجبا ويعلن عن المتغير من هذا النوع بالعبارة int .

مثال 1-10) المثال التالي يوضح أن المعرفات c,b,a هي متغيرات من النوع الصحيح القصير(short)

```
int a,b,c;
```

في حين الاشهار

```
long id_number;
```

يعني ان المتغير id_number هو من النوع الصحيح الطويل (long) وله سعة كبيرة ومداه أكبر من العدد الصحيح القصير (short).

مثال 1-11) المثال التالي يبين الاعلان عن المتغيرات k,j,i من النوع الصحيح int مع تخصيص قيم صحيحة لكل واحد منها.

```
int i=500000;
int j=500000;
int k=i+j;
```

لو تم وضع هذه الجمل في داخل برنامج مع امر الطباعة وبالتالي تنفيذه سنحصل على قيم للمتغيرات مشابهة للآتي:-

I=-24288 J=-24288 K=16960

نلاحظ هنا ان الناتج غير صحيح، وذلك لان كلا من المتغيرات k , j , i تم اشهارها على انها متغيرات صحيحة من النوع int أي القصير، وعليه لايمكن تحميل قيمة كبيرة فيها، وهذا ينتج عنه عدد فائض (integer overflow)، ولتصحيح هذا الخطأ يجب الاعلان عن المتغيرات من النوع الطويل (long) كما يلي:-

```
long i=500000;
long j=500000;
long k=i+j;
```

في هذه الحالة يكون ناتج ما خزن في المتغيرات الثلاثة هو الآتي:

I=500000 J=500000 K=1000000

3) المتغيرات الحقيقية Float Variables

وهي متغيرات تحتوي على نقطة عشرية، أي العدد الذي به قيمة كسرية وينبغي الاعلان عن المتغيرات الحقيقية بالعبارة (float).

مثال 1-12) المتغيران B , A من النوع الحقيقي وخصصت لهما القيم 500 و 300 كالآتي:

```
float A,B;
A=500;
B=300;
```

بينما حاصل ضرب هذين المتغيرين خصص للمتغير الحقيقي C

```
float C=A*B;
```

هنا سيكون ما خزن في المتغير C هو القيمة 150000

فاذا ما تم تخصيص قيم اخرى للمتغيرين B , A كالآتي :-

```
A=500000;
B=1000000;
```

وفي حالة اخراج ما خزن في المتغيرات الثلاثة سيكون الناتج مشابها للآتي:-

```
A=500000
B=1000000
C=5e+11
```

4) المتغيرات مضاعفة الدقة Double Variables

وهي تلك المتغيرات التي تحوي أعدادا صحيحة أو حقيقية ولكنها مضاعفة أو دقيقة جدا، ويعلن عن هذا النوع بكلمة (double).

مثال 1-13) المعرفات z,m,f هي متغيرات مضاعفة الدقة

```
double f,m,z;
```

ايضا يمكن الحصول على نفس النتيجة في المثال الاخير، وذلك باشهار المتغيرات C , B , A لتصبح من النوع المضاعف الطويل كالاتي:-

```
long double A,B,C;
```

من حين الى آخر قد يحتاج المبرمج الى تحويل قيم نوع بياناته اثناء قيامه بتنفيذ برنامجه، وهذا جائز باستخدام صيغة التحويل التالية:-

```
Type(expression)
```

حيث:

Type نوع البيان المراد تحويله

expression أي تعبير

مثال 1-14) على فرض ان لدينا الاعلان التالي :-

```
double d=100.9;
```

واراد المبرمج تحويل قيمة المتغير d الى قيمة من النوع الصحيح ، عليه فان الامر

```
int (d);
```

سوف يقوم بذلك لتصبح قيمة الناتج تساوي 100

7.1) التعليقات Comments

وهي عبارة عن بعض الأوامر الايضاحيه والتوضيحية ينظر القارئ اليهـا وكأنها مرشد ولا يكون لها أي تأثيرفي البرنـامج، لانهـا لا تعتبر جـزءا مـن البرنامج وتستخدم لتسهيل اعادة قراءة البرنامج أو تعديله من طرف المبرمج.

تستخدم التعليقات أيضا لشرح وبيان السبب وراء أي شيء يفعله المـبرمج داخل برنامجه، وتوضع في أي مكان من البرنامج ومن الممكـن أن لا تكـون موجودة في البرنامج، ويبدأ التعليـق باسـتخدام العلامتيـن (//) وينتهـي بنهايـة السطر، اما في حالة ان التعليق في اكـثر مـن سـطر، فمـن المناسـب ان يبـدأ التعليق بالرمزين (*/) وينتهي بالرمزين (/*).

مثال 1-15) بعض التعليقات المقبولة

```
// This is comment statement using C++
// ##### The following program calculates the
//              average of student grades #####
```

مثال 1-16) يمكن اعادة كتابة التعليقات بالمثال السابق على النحو التالي :-

```
/* This is comment statement using C++
   ##### The following program calculates the
   average of student grades ##### */
```

8.1) تمارين Exercises

(1) اشرح ما هو المقصود بالآتي :-

(a) Declaration

(b) Character

(c) Identifier

(d) Variable

(2) الق نظرة على الثوابت التالية وحدد المقبول منها مع ذكر الاسباب.

(a) ox15	(b) 24	(c) 123-45-6789	(d) '9/3'
(e) -456	(f) +1	(g) 0.126e3	(h) .000500
(i) oxde	(j) 8:4	(k) 5.2x6.3	(l) 56%
(m) o12	(n)	(o) 9/3	(p) 1,5

(3) الق نظرة على المعرفات التالية وحدد غير المقبول منها مع ذكر السبب.

(a) dollar$andcents	(b) K : 66	(c) int	(d) to day
(e) a5	(f) 5a	(g) nice-to-meet-you	
(h) first name	(i) Double	(j) whynot?	(k) _tax_rate
(l) C++book	(m) student.name		(n) integer

(4) أي من الآتي لا يعتبر من النوع السلسلة (string)

(a) "Nice-to-see-you "	(b) byte"	(c) "Don't"
(d) '5' `	(e) ' string'	(f) "Z"
(g) "Don\t"	(h) ""$25.9""	(i) "MS-DOS'
(j) "x\n"	(k) "10-6=4"	(l)"Fortran"

(5) عرف العدد مع ذكر انواعه واعطاء بعض الامثلة عنه .

(6) ما المقصود من استخدام الكلمة comment واين يكون موقعها في البرنامج مع اعطاء بعض الامثلة عن ذلك.

(7) وضح الفرق بين كل مما يلي :-

(a) double & float

(b) long & short

(c) string & character

(d) identifier & variable

(8) اعط بعض الامثلة عن الفقرات في تمرين (7).

(9) المطلوب كتابة الاعلان المناسب للمعرفات التالية بحيث تتمشي مع القيم المخصصة لكل معرف حسب الجدول التالي:-

القيمة المخصصة	المعرف
765.95	price
2000	year
100k/h	speed
+	plus
CS115	code
123456789	tolong
raif bashir	friend

(10) بين الخطأ فيما يلي ان وجد مع ذكر السبب :-

(a) /* This is an example */ // using /* Comment Statement */

(b) /* This is an example */ // using /* Comment Statement

(c) // This is an example */ using /* Comment Statement *//

(d) // This is an example // using Comment Statement //

(e) // This is // a program /* number one */

(11) المطلوب إشهار وتخصيص القيم 56.95 , hi user , 20094567 للمتغيرات c3, c2, c1 على التوالي .

(12) اذكر الاخطاء مع تصحيحها أن وجدت :-

(a) INT A, B

(b) float ali, sky;

(c) char name = basher;

(d) short student id;

(e) double float_number;

(f) int K = 12345678;

(g) char ch = 'R';

الفصل الثاني: قنوات ادخال واخراج البيانات

هذا الفصل يبين شرح مجموعة من البرامج البسيطة التي يتم فيها ادخال البيانات بمختلف انواعها، وحتي يكون البرنامج مفهوما لمن كتبه ولغيره من المستخدمين، ينبغي ان تكون طريقة ادخال البيانات ومن ثم اخراجها مفهومة وبدون تعقيد.

1.2) شكل البرنامج Program Style

البرنامج هو عبارة عن مجموعة من الاوامر والتعليمات المكتوبة بشكل منطقي ومتسلسل من قبل المبرمج أو مجموعة من المبرمجين، كل امر أو تعليمة من هذا البرنامج هي توجيه لجهاز الحاسب الآلي (Computer) لاداء عملية معينة من المسألة المعطاة، وعند تنفيذ هذا البرنامج يتم ترجمتة الى لغة يفهمها جهاز الحاسب وهي لغة الآلة (Machine Language) عن طريق المترجم (Compiler) وبالتالي كشف وتسجيل الاخطاء الواردة فيه وتبليغ المبرمج بهذه الاخطاء.

والبرنامج في لغة سي++ قد يأخذ الشكل العام التالي :-

```
<header files>        ملفات العناوين
main()
{
      Variable_Declarations      اعلانات داخلية
```

```
        statement _1;
        statement _2;
            . . . .
        statement_Last;
        return 0;
}
```

مثال 1-2) البرنامج الموالي يقدم تركيب وشكل البرنامج في لغة سي++ مع شرح لمكوناته وهو كما يلي:-

```
// The following is program #1
#include <iostream.h> // for cout
void main(void)
{
        cout<<" C++ is a good language to learn ";
        return 0;
}
```

اذا ما نفذ هذا البرنامج سينتج عنه ظهور السطر التالي:-

C++ is a good language to learn

كما نلاحظ في السطر الاول من هذا البرنامج توجد به الجملة التوضيحية التالية:-

The following is program #1

يسبقها الرمزان (//) وهذ يعني ان الجملة هي جملة تعليق على البرنامج حيث يتجاهلها المترجم، يلي ذلك السطر الثاني وبه الامر

#include <iostream.h>

والذي يسمح للبرنامج باستعمال قناة الادخال istream وقناة الاخراج ostream الموجودتين في ملف العنوان header file تحت اسم iostream.h.

أما السطر الثالث فتوجد به الدالة الرئيسية main والتي يبدأ بها أي برنامج مع ظهور void داخل قوسين للدلالة على عدم وجود اية ادلة أو عوامل لهذه الدالة، يتبع هذه الدالة القوس المفتوح } والذي يدل على بداية جمل البرنامج.

أما قناة هدف الاخراج cout فهي تقوم بارسال الرسالة الموجودة بين علامتي التنصيص (") وطباعتها على شاشه العرض، مع ملاحظة ان كل جملة في لغة سى++ يجب ان تنهي بالفاصلة المنقوطة (;) يلي ذلك جملة (return) والتي تدل على نهاية البرنامج نهاية ناجحة وعادة ما ترجع في مثل هذه الحالة بالقيمة 0، اخيرا من الضروري أن ينتهي البرنامج بالقوس المغلق {للدلالة على نهاية الدالة الرئيسية وبالتالي نهاية البرنامج.

2.2) قنوات الادخال والاخراج Input Output streams

تعتمد لغة سى++ في تصميم أدوات الادخال والاخراج على اساس القنوات (streams) التي يمكن توصيلها باحد الاطراف مثل لوحة المفاتيح والشاشة والطابعة وغيرها ، عن طريق مكتبة قنوات الادخال والاخراج (iostream) من خلال شجرتين من الفصائل ، الفصيلة الاولي (streambuf) وبها الادخال والاخراج غير المشكلة (Unformatted) أما الثانية فهي ios وبها الادخال والاخراج المشكلة (Formatted) ويتفرع منها :

- فصيلة (iostream) لعمليات الادخال والاخراج
- فصيلة (istream) لعمليات الادخال
- فصيلة (ostream) لعمليات الاخراج

3.2) هدف قناة الاخراج Output Stream Object

ان هدف قناة الاخراج (output stream object) وهي cout والتي تدخل ضمن الملفات الرئيسية (header files) في مكتبة الادخال والاخراج (iostream) عن طريق ملف العناوين <iostream.h> تستخدم في عملية الاخراج على وحدة الاخراج القياسية (Standard Output Unit) شاشة العرض (screen) أو الطابعة أو غيرها من ادوات الاخراج.

تأخذ قناة الاخراج الشكل التالي :-

```
cout << Expr1 << Expr2 << ... << Exprn;
```

الاسهم المزدوجة >> والمتجهة الى اليسار تعني مؤثر قناة الاخراج (output stream operator) ويستخدم جنبا الى جنب مع cout وهو يجبر ارسال الاشياء التي تظهرعلى يمينه الى أي شيء يظهر على يساره وتتميز هذه المؤثرات بانها ذكية لانها تستدل على نوع البيانات المطلوب طباعتها بمختلف انواعها تلقائيا.

أما Expr1,Expr2,..., Exprn فهي تعبيرات يمكن ان تكون ثوابت عددية أو قيم لمتغيرات أو تعبيرات من النوع الصحيح أو النوع الحقيقي أو الحرفي المطلوب اخراجها على شاشة العرض.

مثال 2-2) خذ مثلا الجملة

```
cout<<number;
```

تعني حرك أو ارسل قيمة المتغير (number) الى شاشة العرض مثلا، أو بمعني آخر فان الاسهم تخرج من المتغير (number) متجهة الى قناة الاخراج cout وهي الطابعة أو الشاشة أو غيرها.

هناك بعض رموز الحروف الخاصة ، ويطلق عليها احيانا حروف الهروب (Escape Characters) حيث تستخدم لاغراض خاصة مع قناة الاخراج cout ، للتحكم في المخرجات على شاشة العرض ، وتبدأ هذه الرموز بالخط المائل (\) ، على أن يكون ضمن علامة التنصيص المزدوجة (") والجدول التالي يتضمن بعض هذه الرموز :-

اسم الرمز	كيفية كتابته
القفز الى سطر جديد	\n
التقدم 7 مسافات عمودية قبل الطباعة	\t
مسافة الى الخلف	\b
صفحة جديدة	\f
استخدام الجرس	\a
لطباعة الحرف \	\\
لطباعة علامة ؟	\؟
لطباعة علامة التنصيص "	\"
لطباعة علامة التنصيص '	\'
رمز نهاية السلسلة	\0

مثال 2-3) على فرض ان لدينا الجملة التالية :-

```
cout << "The sum of" << 10<<"+"<<15<<" is "<<10+15;
```

فهي تسبب في ارسال وطباعة السطر الموالي :-

```
The sum of 10+15 is 25
```

مثال 2-4) البرنامج الموالي يحتوي على اكثر من قناة اخراج

```
#include<iostream.h>  // for cout
void main(void)
{
       cout<<"My name is RAEF";
       cout<<"here name is MIRATH";
       cout<<"what is yours please?";
       return 0;
}
```

وعند تنفيذه سيظهر على الشاشة النتائج المشابهة للآتي:-

My name is RAEFhere name is MIRATHwhat is yours please?

والذي يلاحظ انه بالرغم من وجود ثلاث جمل اخراج، فان الناتج قد ظهر في سطر واحد، هذا من ناحية، اما من ناحية اخرى فلا يوجد فراغ بين مخرجات هذه الجمل الثلاثة.

وحتي يكون الناتج واضحا، يمكن اعادة كتابة البرنامج السابق بترك فراغ في آخر أو في بداية كل عبارة.

مثال 5-2) المطلوب اعادة كتابة البرنامج في المثال السابق بحيث يكون الناتج كل جملة في سطر منفصل.

```
#include<iostream.h>
void main(void)
{
       cout<<"My name is RAEF \n";
       cout<<"here name is MIRATH \n";
       cout<<"what is yours please?";
       return 0;
}
```

للحصول على المطلوب، يجب استخدام رمز الهروب والخاص بالانتقال الى سطر جديد، وهو (\n) وذلك بوضعه داخل علامة التنصيص المزدوجة، وعليه سيطبع العبارة الاولى وهيMy name is RAEF في بداية السطر الاول، وحيث أن هذه العبارة تنتهي بالرمز (\n) والتي ينتج عنها القفز الى سطر جديد قبل كتابة الجملة الموالية ومن ثم تطبع العبارة here name is MIRATH في بداية السطر الثاني، وهكذا ستطبع العبارة الاخيرة what is yours please? في السطر الاخير.

عموما سيكون ناتج تنفيذ البرنامج السابق كالآتي:-

```
My name is RAEF
here name is MIRATH
what is yours please?
```

قد تكتب جمل البرنامج السابق في امر واحد بالشكل التالي:-

```
cout<<"My name is RAEF \nhere name is MIRATH \nwhat is yours please? ";
```

أو بالشكل التالي:-

```
cout <<"My name is RAEF \nhere name is MIRATH"
     <<"\n what is yours please? ";
```

لاحظ عدم نهاية السطر الاول بالفاصلة المنقوطة.

4.2) تشكيل المدخلات والمخرجات Input Output Format

تقدم لنا لغة سى++ عدداً من ادوات التشكيل (I/O Manipulators) تستخدم في ادخال واخراج البيانات التي تساعد المبرمج على التحكم في مدخلات

ومخرجات برامجه بالطريقة الملائمة والتي يريدها عن طريق مجموعة خاصة من الدوال وهي معرفة بملف العناوين <iomanip.h> والجدول التالي يبين هذه الادوات.

اسم الأداة	اتجاه القناة	المهمة
dec	I/O	تحويل الرقم الى النظام العشري
oct	I/O	تحويل الرقم الى النظام الثماني
hex	I/O	تحويل الرقم الى النظام الستة عشري
endl	O	الانتقال الى سطر جديد
ends	O	لحشر لبنة نهاية الحرف NULL
setbase(int b)	I/O	ضبط قاعدة تحويل الاعداد المختلفة حيث b يأخذ القيم 0,8,10,16
setiosflags(long f)	I/O	رفع راية التشكيل
resetioflags(long f)	I/O	خفض راية التشكيل
setfill(char c)	O	تعبئة الخانات بحرف معين
setprecision(int p)	O	ضبط دقة العدد العشري الى عدد p من الخانات للكسر
setw(int w)	I/O	تحديد اتساع الحقل بالعدد w

مثال 2-6) يمكن اعادة البرنامج المكتوب بالمثال (2-5) والحصول على نفس النتائج باستخدام اداة نهاية السطر (end of line) وهي endl التي تعتبر مكافئة للرمز (\n) كما يلي :-

```
#include<iostream.h>
void main(void)
```

```
{
      cout <<"My name is RAEF"<<endl
            <<"here name is MIRATH"<<endl
            <<"what is yours please? ";
      return 0;
}
```

حيث تشير endl الى الحاسب بالذهاب الى سطر جديد على شاشة العرض.

يمكن التحكم في طول أو اتساع الحقل (Field Width) أي تحديد عدد المواقع التي يتم تخصيصها لاية قيمة باستخدام الاداة (setw) أو عضو الدالة (Member Function) وهي (width) وفي حالة ان عدد المواقع المخصصة هي اقل مما ينبغي، عليه لا يتم تقريب تلك القيمة بل تطبع كما هي، اما في حالة ان عدد المواقع اكبر من المطلوب ، هنا تطبع القيمة وتعبأ بقية الخانات بفراغات أو قد تعبأ بحرف معين عن طريق (setfill) أو عضو الدالة (fill).

مثال 2-7) خذ مثلا الجملة

```
cout<<"ONE"<<"TWO"<<"THREE";
```

سينتج عنها عدد من السلاسل ملتصقة مع بعض في سطر واحد على النحو التالي:-

```
ONETWOTHREE
```

وهذا غير واضح للمنفذ.

فاذا ما استبدلت الجملة السابقة بالآتي :-

```
cout<<"ONE"<<setw(7)<<"TWO"<<setw(7)<<"THREE";
```

حيث استخدمت setw مع cout وبالتالي اذا ما وضعت هذه الجملة في برنامج وتم تنفيذه سيكون الناتج هو المشابه للآتي مع ملاحظة ان الحرف b يعني فراغ (Space)

ONEbbbbTWObbTHREE

وكما نلاحظ فانه تمت طباعة السلسلة الاول ONE بداية من العمود الاول وهذا لا خلاف عليه ، يلي ذلك حجز 7 خانات أو مواقع للسلسلة الحرفية TWO بداية من العمود 4 الى العمود 10 وطباعتها في اقصي يمين هذه المواقع، اخيرا تم تخصيص 7 مواقع للسلسة الثالثة THREE حيث تم طباعتها في اقصي يمين مواقع الاعمدة بداية من 13 وحتى 17.

مثال 2-8) الجملة التالية

```
cout<<setfill('*')<<setw(9) <<55<<"Fifty five";
```

مكافئة للجمل التالية باستخدام دالتي العضو fill و width

```
cout.fill ('*');
cout.width(9);
cout<<55;
cout.width(9);
cout<<"Fifty five";
```

وعليه يكون الناتج في الحالتين

```
*******55Fifty five
```

حيث تطبع القيمة 55 وقبلها 7 نجمات ويتبعها السلسلة Fifty five مباشرة.

مثال 2-9) البرنامج التالي مهمته قراءة قيمتين من النوع الصحيح والحقيقي مع طباعتهما باستخدام كل من (setfill) و (setw).

```
#include<iostream.h>   // for cout
#include<iomanip.h>    // for setw
void main()
{
        int n_i=123;
        float n_f=5.6;
        cout <<endl<<"The numbers you have are :";
        cout<<endl<<setw(10)<<setfill('*')<<n_i;
        cout<<endl<<setw(10)<<setfill('*')<<n_f;
        return 0;
}
```

بعد التنفيذ سيرد الحاسب بالاجابة المشابهة وهي

```
The numbers you have are :
*******123
*******5.6
```

حيث تم حجز عدد 10 مواقع لاخراج القيمتين مع تعبئة الخانات الباقية من هذه المواقع بالرمز (*) على يسار القيمتين نتيجة استخدام الاداة (setfill).

كما تستخدم الدالة setbase() للتحويل بين نظم الاعداد المختلفة حيث يأخذ معاملها 8 للنظام الثماني، 10 للنظام العشري، 16 للنظام الستة عشري، اما الرقم 0 فيستخدم مع النظام العشرى في الاخراج فقط.

مثال 2-10) ما هي مهمة البرنامج التالي ؟

```
#include<iostream.h>
#include<iomanip.h> // for setbase
main()
{
        int num=106;
        cout <<"Hi your number is ==>: ";
        cout <<setbase(10)<<num<<endl;
```

```
        cout <<"Your number in octal system ==>: ";
        cout <<setbase(8)<<num<<endl;
        cout <<"Your number in hexadecimal system ==>: ";
        cout <<setbase(16)<<num<<endl;
        return 0;
}
```

الجواب: عن طريق الدالة (setbase) ومعاملها 10 تم ضبط قاعدة تحويل القيمة الصحيحة 106 الى النظام العشري وتحويل نفس القيمة الى النظام الثماني باستخدام المعامل 8 وهكذا، وفيما يلي ناتج تنفيذ هذا البرنامج.

```
Hi your number is ==>:106
Your number in octal system ==>:152
Your number in hexadecimal system ==>:6a
```

يمكن التحكم في اظهار العدد الكسرى الذي يلي الفاصلة العشرية في العدد الحقيقي وضبطه بدقة وتقريبه الى عدد من الارقام العشرية وذلك باستخدام الاداة (setprecision).

مثال 2-11) اليك جملة cout التالية:-

```
cout<<"A="<<8./3.<<endl<<" B="<<3.456<<endl<<"C="<<0.25;
```

تسبب هذه الجملة في اظهار نتائج التعبيرات الثلاثة كالعادة أي طباعة العدد الكسري في ستة ارقام معنوية تقريبا كالآتي:-

```
A=2.666667
B=3.456
C=0.25
```

ولكن اذا ما غيرت جملة الاخراج السابقة لتصبح بالصورة التالية :-

```
cout <<setprecision(1)<<"A="<<8.3.0<<"\nB="<<3.456<<"\nC="<<0.25;
```

حيث تم استخدام (setprecision) وينتج عنها تخصيص خانة واحدة وقت الطباعة للقيم الكسرية التي بعد الفاصلة العشرية في جميع الاعداد، وعليه يكون ناتج هذه الجملة هو الآتي:-

```
A=2.7
B=3.5
C=0.3
```

وحيث انه تم تخصيص خانة واحدة للقيمة الكسرية، فقد تم تقريب كل قيمة كسرية الى رقم واحد فقط.

يمكن استخدام دالة العضو (precision) لتقريب العدد الحقيقي، فالامران

```
cout.precision(2);
cout<<78.476<<endl;
```

ينتج عنهما تقريب العدد الكسري بالرقم 78.476 وهو 476. الى رقمين فقط ليصبح الرقم 78.48

5.2) رايات التشكيل Formatting Flags

وهي عبارة عن مجموعة من الدوال الخاصة معرفة بملف العناوين <iomanip.h> ويطلق عليها ايضا دوال الاعضاء للفصيلة (ios) وتتحكم في عناصر البيانات والمعلومات واخراجها على وحدة الاخراج كما يريدها المبرمج وذلك عن طريق مجموعة من الرايات التي تفصل فيما بينها بالرمز (|) اذا كانت اكثر من واحدة، والجدول الموالي يوضح البعض منها.

اسم الراية	مهمتها
skipws	تخطي المسافات الفارغة (Spaces)
left	ازاحة الاخراج الى اليسار
right	ازاحة الاخراج الى اليمين
dec	تحويل الرقم الى النظام العشري
oct	تحويل الرقم الى النظام الثماني
hex	تحويل الرقم الى النظام الستة عشري
showpoint	استخدام العلامة العشرية والاصفار الزائدة على اليسار
uppercase	استخدام الحروف الكبيرة في الاخراج
showpos	استخدام العلامة + للاعداد الموجبة
scientific	استخدام الطريقة العلمية الأسية (E)
fixed	استخدام طريقة العلامة العشرية
showbase	استخدام القاعدة العددية

مثال 2-12) البرنامج التالي يبين كيفية استخدام الاداة (setiosflags) مع الراية (showpoint)

```
#include <iostream.h>
#include <iomanip.h>
main()
{
      double A=1.0/3,B=3.125,C=6.3456789;
      cout <<setiosflags(ios::showpoint) <<setw(15)<<A<<endl
           <<setw(15)<<B<<endl<<setw(15)<<C;
      return 0;
}
```

هنا تم استخدام (setiosflags) مـع الرايـة (showpoint) للدلالـة علـى طباعـة بقية مواقع العدد الكسري بالاصفار، وهذا يعني طباعة قيم المتغيرات الحقيقية C,B,A في عدد 15 موقعا مع تعبـأة بقيـة المواقـع علـى يميـن العـدد الكسـري بالاصفار وتعبئة بقية المواقع على يسـار العـدد الصـحيـح بالفراغـات، وعليـه ستكون النتيجة مشابهة للآتـي:-

```
bbbbbbb0.333333
bbbbbbb3.125000
bbbbbbb6.345679
```

مثال 2-13) يمكن استبدال الجملة

```
cout<<setiosflags(ios::showpoint);
```

بالجملة

```
cout<<setiosflags(ios:: scientific);
```

وعليه سوف يعطي البرنامج القيم بالصورة الاسية

```
bbb3.333333e-01
bbbbbb3.125e+00
bbb6.345679e+00
```

اما اذا ما استبدلت الجملة الاخيرة

```
cout<<setiosflags(ios:: scientific);
```

باستخدام الراية (left) لتصبح

```
cout<<setiosflags(ios:: left);
```

هنا تتم ازاحة وطباعة كل القيم الى ناحية اليسارعلى النحو التالي:-

```
0.333333
3.125
6.345679
```

مثال 2-14) الجملة الموالية

```
cout<<setiosflags(ios::left)<<setw(9)<<55<<setw(9)<< "Fifty five";
```

تكافيء الجمل التالية باستخدام دوال العضو

```
cout.setf(ios::left);
cout.width(9);
cout<<55;
cout.width(9);
cout<<"Fifty five";
```

وجميعها ينتج عنها طباعة القيمة 55 في اقصي اليسار يتبعها 7 فراغات يليها طباعة السلسلة Fifty five مباشرة كالآتي:-

```
55bbbbbbbFifty five
```

6.2) هدف قناة الادخال Input Stream Object

في بعض الاحيان قد يلجأ المبرمج الى استخدام مؤثر التخصيص (=) لتخصيص قيم مختلفة لمتغيرات مختلفة في برنامجه، كما تم فعله في بعض البرامج السابقة، ولكن هذه الطريقة لا تمكن المبرمج من تغيير تلك القيم الا بتغيير جملة التخصيص حيث تكون ثابتة اثناء تنفيذ البرنامج خصوصا عند ادخال بيانات كثيرة.

من هنا وجب تقديم هدف قناة الادخال (input stream object) وهي cin التي تدخل ضمن الملفات الرئيسية (header files) في قنوات الادخال والاخراج (iostream) عن طريق ملف العناوين < iostream.h > والتي تشبه قناة الاخراج cout حيث انها تستخدم في عمليات ادخال قيم عن طريق لوحة المفاتيح (keyboard) ومن ثم يجري تحريكها وتخصيصها لاسماء متغيرات، لاستخدامها في البرنامج وقت الحاجة اليها.

الشكل العام لهذه الدالة هو

```
cin >> Variable_1 >> Variable_2 >> ...;
```

حيث:

الاسهم المزدوجة >> والمتجهة الى اليمين أي في اتجاه المتغير فهي تعني مؤثر قناة الادخال (input stream operator) ويستخدم جنبا الى جنب مع cinوهي تتميز بانها ذكية لانها تستدل على نوع البيانات بمختلف انواعها تلقائيا.

أما ...,Variable_1, Variable_2 فهي متغيرات لقيم يتم استقبالها عن طريق لوحة المفاتيح من خلال قناة الادخال ومن ثم تخزينها في ذاكرة الحاسب الآلي تحت اسماء هذه المتغيرات.

مثال 2-15) خذ مثلا الجملة

```
cin>>number;
```

تعني حرك البيانات المدخلة عن طريق لوحة المفاتيح في اتجاه المتغير (number)، أو بمعني آخر فان الاسهم تدخل الى المتغير (number) من مصدر الادخال cin، وعليه فاذا ما وضعت هذه الجملة داخل برنامج، فان تنفيذ هذا البرنامج يتوقف منتظرا ادخال القيم المناسبة للمتغير (number) عن طريق وسيلة الادخال وهي لوحة المفاتيح، ومن ثم يجري تخزينها في عنوان المتغير (number) المخصص له في الذاكرة عن طريق هذه القناة.

مثال 2-16) المطلوب كتابة برنامج كامل لادخال ثلاثة قيم من النوع الصحيح ثم ايجاد مجموعها.

```
#include<iostream.h>  //for cin & cout
main()
{
      int a,b,c;
      cin>>a>>b>>c;
      int sum=a+b+c;
      cout<<"A="<<a<<"  B="<<b<<"  C="<<c<<"  SUM="<<sum;
      return 0;
}
```

قناة الادخال cin في هذا البرنامج تبلغ المستخدم لهذا البرنامج لادخال ثلاث قيم عن طريق لوحة المفاتيح بحيث تحول القيمة الاولي الى المتغير a، والقيمة الثانية الى المتغير b، واخيرا تحول القيمة الثالثة الى المتغير c وجميعها من النوع الصحيح.

لهذا يتم ادخال هذه القيم في سطر واحد أو في اكثر من سطر مع فصل عناصر البيانات المدخلة عن بعضها بفراغ أو اكثر، وقد تكتب القيم كالتالي:-

11   22   33

يلي ذلك الضغط على مفتاح الادخال (Enter)، عندها يتم جمع هذه القيم الثلاثة ومن ثم وضع الناتج في المتغير الصحيح (sum) وظهور النتائج بالشكل المشابه في السطر الموالي:-

A=11   B=22   C=33   SUM=66

مثال 2-17) المطلوب كتابة برنامج لتخصيص القيمتين 98 , 76- للمتغيرين من النوع الصحيح i , j والقيمتين 54.0 , 321.3 المتغيرين من النوع الحقيقي b , a واخيرا القيمتين Q , P للمتغيرين من النوع الحرفي x , y مع اخراج جميع هذه المتغيرات، البرنامج التالي يفي بالمطلوب.

```
#include<iostream.h>    //for cin & cout
main()
{
      int j,i;
      float a,b;
      char x,y;
      cin>>i>>j>>a>>b>>x>>y;
      cout <<"I="<<i<<"  J="<<j<<"  A="<<a
           <<"\nB="<<b<<"  X="<<x<<"  Y="<<y;
      return 0;
}
```

ولتحقيق ادخال البيانات، ينفذ هذا البرنامج وبالتالي يتم ادخال تلك القيم بالصورة التالية:-

98 -76 54.0 321.3PQ

أو بالشكل التالي:-

98 -76 54 321.3 PQ

أو بالشكل الآخر

98 -76 54.0 321.3 P Q

حيث نلاحظ أن القيم المدخلة تفصل عن بعضها بفراغ أو اكثر من فراغ، واما بخصوص القيم من النوع الحرفي فلا يجب أن توضع بين علامتي التنصيص الفردية أو الزوجية، وعليه سيكون ناتج تنفيذ البرنامج في كل الاحوال هو المشابه للآتي:-

I=98 J=-76 A=54

B=321.299988   X=P Y=Q

ميزة أخرى تضاف الى لغة سى++ وهو امكانية تحويل العدد الصحيح العشرى الى الثماني أو الى الستة عشرى أو اعادته الى العشرى من خلال قناة الادخال cin عن طريق dec,hex,oct على التوالي.

مثال 2-18) خذ مثلا اذا تم تعريف المتغير n من النوع الصحيح int وتلي ذلك الامر

```
cin >> n;
```

وادخلت القيمة 016 (أي في النظام الثماني) عندها ستعطي n القيمة 14 بالنظام العشري وفي حالة ادخال القيمة 0X16 (بالنظام الستة عشري) عندها ستعطي n القيمة 22 بالنظام العشري.

مثال 2-19) ما هو رد فعل جهاز الحاسب الآلي في حالة ما اذا كانت البيانات المدخلة غير مطابقة للمتغير؟ البرنامج التالي يوضح هذا.

```
#include<iostream.h>
main()
{
	int m;
	float a;
	cin>>m>>a;
	cout<<"M="<<m<<"  A="<<a;
	return 0;
}
```

عند تنفيذ هذا البرنامج وادخال القيمة 55 للمتغير الصحيح m والقيمة 2.5 للمتغير الحقيقي a ، سوف ينتج عنه الناتج الآتي :-

```
M=55    A=2.5
```

وهذه النتائج صحيحة لان القيم المدخلة هي مطابقة من حيث النوع والعدد.

اما في حالة ادخال القيمة 7.9 للمتغير الصحيح m والقيمة 2.5 للمتغير الحقيقي a، فسوف يكون الناتج غير ما يتوقع وهو الآتي:-

```
M=7    A=0.9
```

والسبب هو لان القيمة المدخلة الاولي 7.9 غير مطابقة للمتغير m لذلك تم تخصيص العدد الصحيح لهذه القيمة وهو العدد 7 لهذا المتغير، بينما خصص العدد الكسري لنفس القيمة الاولي وهو 0.9 للمتغير الحقيقي a اما بخصوص القيمة المدخلة الثانية فقد تجاهلها المترجم باعتبارها زائدة.

وبالتالي يجب ان يكون المنفذ لمثل هذه البرامج وخاصة عند استخدام cin على علم مسبقا بعدد المتغيرات ونوعها (حروفا كانت أم ارقاماً أم غيرها)، وحتى يتم اعطاء البيانات بالطريقة السليمة ينبغي طباعة الرسائل المناسبة قبل الوصول الى جملة ادخال البيانات المطلوبة.

مثال 2-20) البرنامج التالي مهمته اظهار رسالة على شاشة العرض قبل الوصول الى قناة cin لتبين وتوضح نوع البيانات المطلوب ادخالها وعددها للشخص المنفذ.

```
#include<iostream.h>
void main(void)
{
	int my_number;
	cout <<"What is your favorite number ? ";
	cin >>my_number;
	cout <<"The number you typed is "<<my_number;
	return 0;
}
```

نلاحظ هنا استخدام قناة الادخال cout الأولى لطباعة الرسالة

What is your favorite number ?

على شاشة العرض بالسؤال عما هو الرقم المحبب اليك ، ويستطيع المنفذ معرفة نوع البيانات المطلوبة لهذا البرنامج، وليكن الرد بطباعة الرقم 7 مثلا

مع الضغط على مفتاح الادخال (Enter) وبالتالي يرد الحاسب عن طريق قناة الاخراج الثانية بالجواب الآتي:-

The number you typed is 7

7.2) دالة ادخال الحرف Input Character Function

نلاحظ عند كتابة البرامج السابقة أنه قد تم استخدام قناة الادخال cin لغرض استقبال البيانات من النوع العددي أو الحرفي، وقد حان الوقت الآن لتقديم دالة أخرى خاصة بادخال الحرف، وهي دالة العضو (member function) وتدعى ()get والتي لها الشكل التالي:-

Input_Stream.get(Char_Variable);

ومهمتها استقبال متغير حرفي واحد Char_Variable عن طريق لوحة المفاتيح.

مثال 2-21) البرنامج الموالي مهمته استخدام هذا النوع من الدوال.

```
#include<iostream.h>
main()
{
        char letter;
        cout<<endl<<"Please enter a letter ==>: ";
        cin.get(letter);
        cout<<endl<<"The letter you typed was ==>: "<<letter;
        return 0;
}
```

نلاحظ انه في هذا البرنامج تم الاعلان عن المتغير (letter) من النوع الحرفي، يلي ذلك طلب الجهاز ادخال الحرف عن طريق الرسالة التالية:-

Please enter a letter ==> :

ويمجرد الضغط على الحرف المطلوب ادخاله وليكن الحرف A مثلا والضغط على مفتاح الادخال (Enter) عندها يتم اسناد هذا الحرف الى المتغير الحرفي letter وطباعته مع الرسالة الموالية مباشرة وهي:-

The letter you typed was ==> : A

مثال 2-22) البرنامج الموالي يبين مهمة أخرى لاستخدام الدالة get ()

```
#include<iostream.h>
main()
{
        cout <<endl<<"Hi user please press any key to"
              <<"return back to main program";
        cin.get();
        return 0;
}
```

نلاحظ في بعض البرامج الماضية أنه عند تنفيذها تظهر النتائج على شاشة العرض ولكن لا يمكن للمستخدم التمعن في نتائجها الا بعد الضغط على المفتاحين Alt-F5 .

اما باستخدام الدالة get() أو الدالة geth() فان البرنامج يعطي فرصة للمستخدم لكي يتمعن في نتائج برنامجه ويستمر هذا الى ان يتم الضغط على أي مفتاح من لوحة المفاتيح عندها يتم الرجوع الى البرنامج الرئيسي.

واذا ما نفذ هذا البرنامج سينتج عنه الرسالة التوضيحة التالية:-

Hi user press any key to return back to main program

وهي تقول للمستخدم اضغط على أي مفتاح للرجوع الى برنامجه.

8.2) دالة اخراج الحرف Output Character Function

هناك دالة مهمتها اخراج الحرف الواحد على شاشة العرض وهي دالة العضو put() وهي المقابلة للدالة get() ولها الشكل:

```
Output_Stream.put(Char_Expression);
```

فمثلا الامر

```
cout.put('B');
```

سينتج عنه طباعة الحرف B.

ودالة put() قليلة الاستعمال نظرا لاستخدام القناة cout في اخراج جميع انواع البيانات عوضا عنها ، ولكن قد تستدعي لاخراج القيمة المقابلة للعدد الصحيح في نظام آسكي (ASCII) كما توضحه الجملة

```
cout.put(65).put('\n').put(97);
```

التي ينتج عنها طباعة الحرف A المقابل للرقم 65 في نظام آسكي في السطر الاول ثم يتم الانتقال الى سطر جديد حيث يطبع الحرف a المقابل للرقم 97.

9.2) ادخال الحرفيات Input Strings

بعد تقديم كيفية التعامل مع المتغير الحرفي character من حيث الادخال والاخراج، فقد حان الآن تقديم كيفية الاعلان عن المتغيرات من نوع السلسلة الحرفية string أي التي تتكون اكثر من حرف أو رمز، وذلك قبل التطرق الى معالجة هذا النوع من المتغيرات.

يتم الاعلان عن المتغير من نوع السلسلة الحرفية باستخدام المؤشر الحرفي الذي مهمته الاشارة الى بداية أي سلسلة حرفية.

الشكل العام:

```
type *variable;
```

فمثلا الاشهار :

```
char *str;
```

يعني ان الرمز (*) يشير الى بداية القيمة المخزنة في العنوان الذي يشير اليه المؤشر str الذي هو متغير من النوع السلسة الحرفية.

خذ مثلا الاعلان

```
char *str="BLUE";
```

المتغير str هو من نوع المؤشر (pointer) يؤشر الى السلسلة "BLUE" في مكان ما في ذاكرة الحاسب.

في حين الاعلان الموالي:-

```
char word[]="BLUE";
```

فيه word متغير من نوع المصفوفة الحرفية يحتوي على 5 عناصر أي عدد حروف كلمة اللون "BLUE" وهي 4 حروف مضافا اليها رمز النهاية ('\0') وفي هذا الاعلان لم يتم تحديد طول ولا نهاية السلسلة، بل المترجم هو الذي يقوم بهذه العملية.

مثال 2-23) ما الذي تلاحظه في البرنامج الموالي؟

```
#include<iostream.h>
main()
{
	char *ch="NICE TO MEET YOU";
	cout<<ch<<endl;
	cout<<*ch;
	return 0;
}
```

الملاحظ هو الآتي :

1) تم تخصيص عنوان بداية السلسلة وهو الحرف N للمتغير ch
2) تمت اضافة الرمز ('0\') في نهاية السلسلة لتحديد نهايتها
3) تبدأ جملة الاخراج الاولي cout بطباعة حروف المتغير ch حرفا حرفا بداية من الحرف N وحتي الوصول الى رمز النهاية ('0\') أي طباعة السطر التالي :-

NICE TO MEET YOU

4) اما جملة الاخراج الثانية سوف تقوم باخراج الحرف الاول N والذي يشير اليه المؤشر فقط .

مثال 2-24) ماذا يحدث اذا ما استخدمت cin لاستقبال سلسلة حرفية ؟ البرنامج التالي يوضح الاجابة.

```
#include<iostream.h>
main()
{
	char * sentence;
	cout<<"Enter your sentence ==>: ";
	cin>> sentence;
	cout<<endl<<"The sentence read with cin was ==> "<< sentence;
	return 0;
}
```

عند ظهور الرسالة

Enter your sentence ==> :

بعد التنفيذ على شاشة العرض ، وعلى فرض أنه قد تم ادخال السلسلة

WHY AREN'T YOU AT WORK TODAY ?

هنا تقوم cin باستقبال السلسلة الحرفية حرفا حرفا حتي تصل الى أول فراغ (space) في هذه السلسة وبالتالي اسناد هذه القيمة الى المتغير sentence وطباعتها عن طريق الرسالة التالية :-

The sentence read with cin was ==>: WHY

10.2) دالة ادخال الحرفيات Input Strings Function

تزودنا لغة سي++ بدالة العضو (member function) وهي getline() التي لها الشكل:

```
Input_Stream.getline(String_Var, Max_Characters+1);
```

ومهمتها التعامل مع السلاسل الحرفيه حتي وأن احتوت على فراغات.

ومن هنا يمكن استخدام قناة الاخراج cout لاظهار السلسلة الحرفية التي قد يتم ادخالها عن طريق الدالة getline() بدون أي مشاكل.

خذ مثلا الامر

```
cin.getline(str, size);
```

وهو يعني نقل أو تحويل السلسلة الحرفية التي ادخلت عن طريق لوحة المفاتيح بعدد محدود من الحروف لا يزيد عن size حرفا بما فيها رمز نهاية السلسلة في آخرها '\n' من قناة الادخال cin وتخزينها في المتغير str.

أو الامر

```
cin.getline (str,size, '\n');
```

حيث القراءة تنتهي بمجرد الوصول الى الرمز '\n'

مثال 2-25) المطلوب اعادة كتابة البرنامج بالمثال (2-24) مستخدما فيه الدالة getline() للادخال مع cout للاخراج.

```
#include<iostream.h>
main()
{
        char sentence[80];
        cout<<"Enter your sentence ==>:";
        cin.getline(sentence,80);
        cout<<" The sentence read with cin.getline was ==>";
        cout<< sentence;
        return 0;
}
```

عند تنفيذ هذا البرنامج ستظهر الرسالة التالية:-

Enter your sentence ==>:

واذا ما تم ادخال الجملة التالية

Why aren't You at Work to Day ?

حينها سوف يتم اسنادها للمتغير (sentence) مع ملاحظة ان السلسلة لا يزيد عدد حروفها على 79 حرفا مضافا اليها رمز النهاية في آخرها، وعليه سيكون ناتج هذا البرنامج هو السطر الموالي:-

The sentence read with cin.getline was ==> Why aren't You at Work to Day ?

اما في حالة استبدال الدالة getline() بالبرنامج السابق لتصبح:-

cin.getline(sentence,8);

عليه سوف يتم طباعة السبعة حروف الاولى من السلسلة الحرفية فقط، ويكون ناتج البرنامج في هذه الحالة ما يلي:-

The sentence read with cin.getline was ==> Why are

11.2) دالة الرفض Ignore Function

ومهمتها تجاهل أو رفض ما هو موجود في قناة الإدخال ، ولكي يتم التعرف على طريقة استخدام هذه الدالة، اليكم هذا المثال .

مثال 2-26) المطلوب كتابة برنامج لإدخال وإخراج رقم قيد الطالب واسمه.

```
#include <iostream.h>
main()
{
        char name[50];
        long id;
        cout<<"Enter your id# ==>";
        cin>>id;
        cout<<"Enter your name ==>";
        cin.getline(name,50);
        cout<<"your id is "<<id<<" and your name is "<<name;
        return 0;
}
```

عند تنفيذ هذا البرنامج ستظهر الرسالة الاولي على النحو التالي :-

Enter your id# ==>

ويتم الرد عليها بإدخال رقم قيد الطالب وليكن 20095432 كالآتي:-

Enter your id# ==>20095432

وبعد الضغط على مفتاح الإدخال (Enter) يرجع التحكم إلى البرنامج الاصلي بدون ان يعطي فرصة لمنفذ البرنامج ليتم إدخال باقي بياناته وهو الاسم name، وبالتالي سيطبع السطر التالي :-

```
Enter your name ==>your id is 20095432 and your name is
```

والسبب هو ان عملية القراءة الأخيرة للمتغير id تترك بعض المخلفات في قناة الإدخال cin، وبالتالي فانه عند الضغط على مفتاح الإدخال (Enter) عندها يتم ارسال علامة السطر الجديد الى قناة الإدخال وتبقي في القناة حتي عملية القراءة التالية، ولان الدالة getline() تقرأ السلسلة الحرفية بما فيها الفراغات فانها سوف تقرأ علامة السطر الجديد كأحد المدخلات وتسبب في ازالة المحتويات الموجودة بالقناة.

لمعالجة هذا الخطأ وحتي يمكن قراءة رقم القيد والغاء أو تجاهل باقي الحروف بالقناة حتي اول سطر جديد لنتمكن من قراءة الاسم، فإننا تستعمل دالة العضو ignore() بالشكل

```
cin.ignore(80,'\n');
```

وتعني ازالة 80 حرفا على الاقل أو عند مصادفة رمز نهاية السطر، وهذا في حالة ان المنفذ لا يريد إدخال الاسم اكثر من 79 حرفا، أو بالشكل

```
cin.ignore(1);
```

حيث حذف حرف واحد من قناة الإدخال وبالتالي فان هذه القناة تكون خالية من أية محتويات.

مثال 2-27) وحتي تكون مدخلات ومخرجات البرنامج السابق صحيحة، يمكن تعديل ذلك البرنامج ليأخذ الشكل الموالي

```
#include <iostream.h>
main()
{
        char name[50];
        long id;
        cout<<"Enter your id# ==>";
        cin>>id;
        cin.ignore(80,'\n');
        cout<<"Enter your name ==>";
        cin.getline(name,50);
        cout<<"your id is "<<id<<" and your name is "<<name;
        return 0;
}
```

المدخلات والمخرجات وقت التنفيذ هي الآتي:-

Enter your id# ==> 20095432

Enter your name ==> SOFYAN SOHIAB GAYED

your id is 20095432 and your name is SOFYAN SOHIAB GAYED

12.2) تمارين Exercises

(1) المطلوب اخراج القيمة 5.96 والقيمة 56.789 والقيمة 34.7286 في جملة واحدة على اساس 3 خانات بعد الفاصلة العشرية.

(2) اكتب الجملة المناسبة لادخال القيمتين 34.5678 , 5678 واظهارهما باستخدام دالة العضو المناسبة بالشكل التالي:-

******5678*******34.568

(3) أجعل الحاسب يطبع الرسالة This is a C++ Book في سطرين بحيث تكون C++ في نهاية السطر الاول.

(4) اكتب برنامجا يكتب الرسالة في التمرين السابق بحيث تكون كل كلمة في سطر منفصل.

(5) اكتب برنامجا مهمته استقبال الاسم name بعد الرسالة التالية:-

Hi, What's your name ==>

ثم اظهاره على النحو التالي:-

Welcome [name]

Let's be Friends !

(6) المطلوب كتابة برنامج كامل مهمته استقبال اسم الطالب وثلاثة قيم تمثل درجاته على النحو التالي:-

Enter student name please ==>: NOWFAL BASHIR

Enter first grade please ==>: 67.5

Enter second grade please ==>: 70

Enter third grade please ==>: 80

ثم اخراج البيانات السابقة مع المتوسط بالشكل الموالي:-

Student name is tarek ali

Grade 1 : 67.5

Grade 2 : 70

Grade 3 : 80

Average : 72.5

(7) أكتب برنامجا يقوم بقراءة سلسلتين ومن ثم طباعتهما بالشكل التالي:-

Hi This is First Program in

C++ Language

(8) اكتب برنامجا يسألك عن اسمك ورقم البطاقة والجنسية مع طباعة رقم البطاقة والجنسية في السطر الاول والاسم والجنسية في السطر الثاني.

(9) اكتب برنامجا لقراءة اسم الطالب Mohammed Bashir ودرجته 83.5 ثم قم باخراج هذه البيانات باستخدام جملة اخراج واحدة كالآتي:-

Student Name ##### Mohammed Bashir ##### has grade 83.5

(10) اذا تم الاعلان عن المتغيرات بالشكل

```
int n1,n2,n3,n4;
double d1,d2,d3,d4;
char c1,c2,c3;
```

فما هي قيم هذه المتغيرات في حالة ادخال البيانات بالصورة التالية حيث b تعني فراغ

12b34E1b-5.67b=89b76b>54

وذلك باستخدام الاجزاء التالية من البرنامج

(a)

```
cin>>n1>>d1>>c1;
cin>>n2>>d2>>c2;
cin>>n3>>d3>>c3;
cout<<n1<<"  "<<d1<<"  "<<c1<<endl;
cout<<n2<<"  "<<d2<<"  "<<c2<<endl;
cout<<n3<<"  "<<d3<<"  "<<c3<<endl;
```

(b)

```
cin>>c1>>n1>>d1;
cin>>c2>>n2>>d2;
cin>>c3>>n3>>d3;
cout<<n1<<"  "<<d1<<"  "<<c1<<endl;
cout<<n2<<"  "<<d2<<"  "<<c2<<endl;
cout<<n3<<"  "<<d3<<"  "<<c3<<endl;
```

(11) أكتب برنامجا كاملا لقراءة اطوال ضلعي مستطيل m,l ثم احسب محيطه V ومساحته A.

(12) اكتب برنامجاً لادخال القيم 56 ، 55 ، 5 ، 6 واظهارها بالشكل التالي:-

The sum of [56] and [55] ==>111

The product of 5 Times 6 is equal to ==>30

(13) في حالة اشهار المتغير n من النوع الصحيح int وتخصيص القيمة 23، المطلوب كتابة وتنفيذ برنامج لكل جزء من الاجزاء التالية مع توضيح الفرق بينها من حيث المخرجات.

(a)

```
cout <<n<<"  in octal is "<<setiosflags(ios::showbase)<<oct<<n<<endl
     <<dec<<n<<"  in hexadecimal is "<<hex<<n<<endl
     <<dec<<n<<"  in decimal is "<<dec<<n<<endl;
```

(b)

```
cout <<n<<"  in octal is "<<oct<<n<<endl
     <<dec<<n<<"  in hexadecimal is "<<hex<<n<<endl
     <<dec<<n<<"  in decimal is "<<dec<<n<<endl;
```

(14) ما هو ناتج تنفيذ الفقرات التالية:-

(a)

```
int num=4321;
cout<<num<<endl;
cout.width(7);
cout.fill('*');
cout<<99<<setw(5)<<setfill('$')<<333;
```

(b)

```
float f=123.68;
cout<<setprecision(3)<<f;
cout<<setw(10)<<setprecision(1)<<f;
```

(c)

```
int n=123;
float f=456.78123;
cout<<setiosflags(ios::showpos)<<n<<endl;
cout<<setiosflags(ios::scientific)<<f;
```

(d)

```
double a=5.64,b=12.183,c=44.99234;
cout <<setiosflags(ios::fixed) << setprecision(1)
     <<"A="<<a<<" B="<<b<<"  C="<<c<<endl;
a=34.876412,b=12345.123,c=456.9321;
cout <<resetiosflags(ios::fixed) << setiosflags(ios::scientific)
     <<setprecision(3)<<"A="<<a<<"  B="<<b<<"  C="<<c<<endl;
a=(float)8/3,b=15/5,c=11/(double)5;
cout << setiosflags(ios::fixed | ios::showpoint)
     <<"A="<<a<<" B="<<b<<"  C="<<c<<endl;
```

(15) اكتب برنامجا مهمته الحصول على الاشكال التالية :-

```
(a) *                (b) **********        (c)       *1*
    **                    *********                *11211*
    ***                    ********              *111131111*
    ****                    *******            *1111114111111*
    *****                    ******           *111111151111111*
    ******                    *****           *111111151111111*
    *******                    ****            *1111114111111*
    ********                    ***              *111131111*
    *********                    **                *11211*
    **********                    *                  *1*
```

(16) اكتب برنامجا كاملا يقرأ درجة الحرارة بالفهرنهية F ثم يطبعها بالمئوية C باستخدام المعالة التالية :-

$$C = 5/9(F-32)$$

(17) المطلوب كتابة برنامج لحساب المعادلة التالية :-

$$X = 3A + 2B - 10$$

(18) اكتب برنامجا كاملا يتم فيه إدخال رقم الطالب واسمه ودرجته مع إخراج هذه البيانات بالصورة التالية :-

STUDENT REPORT

ID NUMBER	NAME	GRADE
20098090	ALI MOHAMMED	88.6

الفصل الثالث: الجمل والمؤثرات

هذا الفصل يتم فيه شرح التعابير والجمل بانواعها مع كيفية معالجة البيانات المختلفة باجراء بعض العمليات عليها باستخدام المؤثرات الحسابية والمنطقية التي تشتهر بها لغة سي++ والتي تعتبر مدخلا لكتابة البرامج التي تعتمد على ادخال ومعالجة واخراج البيانات وتوضيح أولويات تنفيذ هذه المؤثرات واخيرا تقديم بعض الدوال الرياضية التي قد يحتاجها المبرمج.

1.3) التعبيرات Expressions

التعبير يمكن أن يكون قيمة ثابتة مثل 66.7 أو متغيرا مثل total أو متغيرا من نوع المصفوفة مثل matrix[5] أو مجموعة من القيم والمتغيرات متصلة مع بعضها بالمؤثرات (Operators).

1) التعبيرات الحسابية Arithmetic Expressions

وهي عبارة عن مجموعة من المتغيرات والثوابت وتكون نتيجتها قيمة حسابية اما صحيحة أو حقيقية.

خذ مثلا التعبير

(7/2)*m/n+(k % j)

حيث n,m,k,j متغيرات و 2,7 ثوابت في حين *,/,%,+ مؤثرات حسابية.

2) التعبيرات المنطقية Logical Expressions

تتكون من عنصر منطقي واحد أو مجموعة من العناصر المنطقية والعلاقات المنطقية، حيث تعطي احدى القيمتين المنطقيتين اما خطأ (false) أو الرقم (0) واما صوابا (true) أو الرقم (1) أو أي رقم آخر ماعدا الرقم (0).

مثال 3-1) الآتي هو لتعبيرين منطقيين

```
Initial != '? '
(m>= 0) && (n/2>x)
```

2.3) الثوابت Constants

يستخدم الثابت في لغة سي++ حيث يعلن عنه عن طريق الامر const في بداية البرنامج مرة واحده وذلك قبل استعماله، والصيغة العامة للثابت هي:

```
const  Type VariableName=Constant;
```

حيث:

const هي كلمة محجوزة لتعريف الثابت.

Type تدل على نوع الثابت.

VariableName هو معرف يمثل اسم الثابت.

Constant هو عنصر البيانات أو قيمة الثابت الفعلي الذي يحدد له المعرف (VariableName).

يجب الأخذ في الاعتبار أن الثابت يحمل قيما لا يمكن تغييرها في البرنامج بالزيادة أو النقصان أو القراءة.

الامر

```
const float total_amount =5060.55;
```

يعني استبدال المعرف total_amount كلما وجد في البرنامج بالقيمة 5060.55، وعليه لا يمكن تغيير قيمة هذا المعرف اثناء تنفيذ البرنامج، فمثلا من الخطأ كتابة الامر التالي:-

```
total_amount=total_amount+amount;
```

مثال 3-2) فيما يلي بعض الامثلة المقبولة على الثابت

```
const float Speed_of_Light=299792.0;
const char Slash='\';
const char letterGrade='A';
const int DaysPerWeek=7;
```

مثال 3-3) البرنامج التالي يبين كيفية استعمال الأمر const

```
#include<iostream.h>
main()
{
      const int A=0xf;   // Value of A in Hexadecimal number
      float price=25.75;
      char msg[]=" cost me obout ";
      char book_name[]="The C++ BOOK";
      cout<<"Value of A in DECIMAL ======>: "<<A<<endl;
      cout<<"Value of A in Octal   =========>: "<<oct<<A<<endl;
      cout<<"Value of A in HEXADECIMAL ==>: "<<hex<<A<<endl;
      cout<<book_name<<msg<<price<<" dinar"<<endl;
      return 0;
}
```

حيث خصصت القيمة الحقيقية 25.75 للمعرف الثابت price وخصصت للمعرف الثابت book_name السلسلة الحرفية "The C++ BOOK" كما اسندت السلسلة الحرفية "cost me obout" للمعرف الثابت msg في حين ان الرقم oxf

يعتبر في النظام الستة عشري (أي يكـافيء الرقـم 15 فـي النظـام العشـري) أسند للثابت A وسيأتي شرح هذه الانظمة وغيرها فيما بعد ان شاء الله.

البرنامج اذا ما نفذ سوف يعطي الناتج المشابه للآتي:-

```
Value of A in DECIMAL =======>: 15
Value of A in Octal  =========>: 17
Value of A in HEXADECIMAL ==>: f
The C++ BOOK cost me about 25.75 dinar
```

3.3) الجمل Statements

الجملة تعنـي الاشـارة أو الامـر الموجـه الـى الحاسـب لعمـل شـيء معين مطلوب القيام به وتكتب الجملة في لغة سي++ امـا فـي سـطر واحـد، أو فـي اكثر من سطر، وتكون اما مفردة وهي التي تتكون من جملـة واحـدة أو أمـر واحد، حيث يجب أن تنتهي هذه الجملة بالفاصلة منقوطة، مثل

```
a=b+c/d;
```

أو

```
cout<<" grade= "<<grade;
```

أو تكون جملة مركبة وهي التي تتكون من أكثر من جملة مفردة، على ان تحصر هذه الجمل جميعها بين قوسي الفئة {}، مثل

```
{
    cin>>EmpNumber>>Hours >> Rate;         // Read employee data
    Wages=Hours * Rate;                    // Calculate Wages
}
```

وسيتم شرح هذا النوع من الجمل بعـون اللـه لاحقـا، وتنقسـم الجمـل الـى الآتي:-

أولا)) جملة التخصيص Assignment Statement

وهي تأخذ الشكل العام التالي :-

VariableName = Expression ;

حيث:

VariableName الواقع على يسار رمز التخصيص (=) يجب ان يكون متغيرا تم الاعلان عنه مسبقا.

Expression هو نوع من التعابير الاولية أو المعقدة أو الثابتة أو المتغيرة.

أما اشارة التخصيص (=) فتعني وضع القيمة النهائية الموجودة في الطرف الايمن من هذه الاشارة وحفظها في المتغير الموجود في الطرف الايسر منها.

مثال 3-4) فيما يلي بعض الامثلة على التخصيصات المقبولة:-

pi=3.141519 ;	قيمة ثابتة
Result=Total ;	متغيرة
H = R+2.3* W ;	تعابير أولية
n=(q*(b*k+m))/f ;	تعابير معقدة

في حين أن الامثلة التالية غير مقبولة، وذلك للسبب المدون قرين كل منها.

a + b = 55.5 ;	لا يمكن تخصيص قيمة 55.5 لتعبير a + b
-m=300 ;	لا يسمح باشارة (-) في الطرف الايسر
x1=-b+(b*b-4.0*a*b))/(2*a) ;	الاقواس غير كاملة

خذ مثلا جملة التخصيص التالية:-

```
int sum=sum+2 ;
```

تعني ان المتغير sum من النوع الصحيح واذا كانت له القيمة 10 مثلا، فان قيمته سوف تساوي 12، وهي تعني جمع العدد 2 على القيمة القديمة sum واسناد النتيجة الى نفس المتغير.

مثال 3-5) لنفرض أن لدينا المتغيرات C,B,A من النوع الحقيقي وتم تخصيص 5.55 و 7.77 الى المتغيرين B,A على التوالي، والمتغير C هو حاصل جمع المتغيرين B , A وعلى هذا سيكون البرنامج كالآتي:-

```
#include <iostream.h>
main()
{
	float A=5.55;
	float B=7.77;
	float C=A+B;
	cout<<"\n A="<<A<<"  B="<<B<<"  C="<<C;
	return 0;
}
```

وعند التنفيذ سوف يعطي هذا البرنامج النتائج التالية:-

```
A= 5.55   B=7.77   C=13.32
```

ايضا يمكن استخدام علامة التخصيص (=) في اكثر من موقع في التعبير الواحد.

مثال 3-6) يمكن اعادة نفس البرنامج في المثال السابق باستخدام اكثر من علامة تخصيص واحدة وذلك على النحو التالي:-

```
#include <iostream.h>
main()
{
        float A,B;
        float C=(A=5.55)+(B=7.77);
        cout<<"\n A="<<A<<" B="<<B<<" C="<<C;
        return 0;
}
```

الناتج سوف يكون مشابها للناتج في المثال السابق.

ثانيا)) جملة التخصيص الممتدة Extended Assignment Statement

المترجم في لغة سى++ يسمح باستخدام جملة التخصيص الممتدة، وهي نادرا ما تجدها في اللغات الاخرى، بحيث يمكن الشخص المبرمج من تخصيص قيمة معينه لاكثر من متغير في نفس الوقت وتخزينها في الذاكرة تحت اسماء تلك المتغيرات، وتأخذ هذ الجملة الشكل التالي:-

Variable1=Variable2=... = Expression ;

مثال 3-7) اليك جمل التخصيص التالية:-

x=1234;   y=1234;   z=1234;

نلاحظ هنا أنه قد تم استخدام ثلاثة جمل تخصيص وذلك لاسناد القيمة الصحيحة 1234 لكل من المتغيرات z,y,x على التوالي، وهذه طريقة غير مناسبة خصوصا عند اسناد عدد من المتغيرات لقيمة واحدة.

مثال 3-8) البرنامج المولي مهمته تخصيص القيمة 1234 لعدد من المتغيرات الصحيحة والقيمة الحرفية '?' لعدد من المتغيرات الحرفية باستخدام جملة التخصيص الممتدة.

```
#include <iostream.h>
main()
{
        int x,y,z;
        char a,b,c;
        x=y=z=1234;
        a=b=c='?';
        cout<<"\n X="<<x<<"  Y="<<y<<"  Z="<<z;
        cout<<"\n A="<<a<<"  B="<<b<<"  C="<<c;
         return 0;
}
```

هنا يبدأ تخصيص القيمة 1234 من اليمين الى اليسار، أي تخصيص وتخزين القيمة 1234 الى المتغير z أولا، ثم تخصيص القيمة المخزنة في المتغير z الى المتغير y ، وأخيرا تخصيص قيمة y الى المتغير x، وبنفس الطريقة يتم اسناد القيمة الحرفية علامة الاستفهام ? للمتغيرات c,b,a وعلى هذا سوف يكون ناتج هذا البرنامج عند تنفيذه كالآتي:-

```
X=1234    Y=1234   Z=1234
A=?   B=?   C=?
```

خذ كمثال الجمل التالية:-

```
int p,q,r=5;
q=2+r+7;
p=q;
```

يمكن كتابتها على النحو التالي :-

```
int p,q,r=5;
p=q=2+r+7;
```

هنا يتم تقويم التعبير 2+r+7 حيث ينتج عنه القيمة 14 ثم اسناد هذه القيمة الى المتغير q أولا والمتغير p ثانيا.

4.3) المؤثرات الحسابية Arithmetic Operators

هناك خمس مؤثرات حسابية ثنائية مألوفة في لغة سي++ تستخدم مع الاعداد هي:-

المؤثر	معناه
+	الجمع
−	الطرح
*	الضرب
/	القسمه
%	قسمة ينتج عنها الرقم الصحيح الممثل لباقي عملية القسمة

ملاحظة: عند تنفيذ أي تعابير حسابية يستخدم فيها المؤثرات الحسابية السابقة، يجب ان تنفذ بالترتيب التالي:-

الاقواس ان وجد والضرب (*) والقسمة (/) وباقي القسمة (%) واخيراً الجمع (+) والطرح (-) وكلهم من اليسار إلى اليمين.

كما يجب أن يؤخذ في الاعتبار في حالة استخدام مؤثر القسمة (/) ان الناتج سيكون عددا صحيحا في حالة ما اذا كانت عناصر البيانات المستعملة أعداداً صحيحة، ايضا بالنسبة للمؤثر (%) يجب أن تكون عناصر البيانات قيما صحيحة والا فالنتيجة ستكون خاطئة.

مثال 3-9) المثال التالي يبين استخدام المؤثرات الحسابية مع المتغيرات الصحيحة والحقيقية كما يلي:-

```
#include <iostream.h>
main()
{
	int A=10, B=5;
	float X=9.6, Y=0.3;
	cout<<"X+A="<<X+A<<"  A-Y="<<A-Y<<endl;
	cout<<"X*Y="<<X*Y<<"  X/Y="<<X/Y<<endl;
	cout<<"X%B="<<X/B<<"  B*A/(B-1)+X/2="<<B*A/(B-1)+X/2;
	return 0;
}
```

ناتج تنفيذ البرنامج السابق سيكون مشابها للآتي:-

```
X+A=19.6   A-Y=9.7
X*Y=2.88   X/Y=32
X%B=1.92  B*A/(B-1)+X/2=16.799999
```

مثال 3-10) فيما يلي الاعلان عن بعض المتغيرات الصحيحة مع جدول يبين عددا من العمليات الحسابية حسب اولويات تنفيذها على جهاز الحاسب.

```
int a=2,b=-5,c=9,d=-17;
float x=4.5,y=3.9;
```

التعابير	القيمة
- - -c	-9
c/b/a	0
c%a	1
a%c	2
d/b%a	1
int (x+y)/float(a+1)	2.666667
a%-b*c	18
-d%(a*c)	17

التعابير	القيمة
-d%a*c	9
c/(5+b)	divide error
double (a-b+d)/int(x)/a	-1.25
d*a%((b+d)/a)*a-d	15

مثال 3-11) المطلوب كتابة برنامج كامل لايجاد مساحة مثلث وهي نصف القاعدة في الارتفاع.

```
#include <iostream.h>
#include <iomanip.h>
      // This program computes area of triangle
      // one-half base times height
main ( )
{
      cout <<"\nPlease enter base of triangle ==>: ";
      float base;
      cin>> base;
      cout <<"\nPlease enter height of triangle ==>: ";
      float height;
      cin  >> height;
      float   area=base * height/2;   // compute the area of triangle
      cout <<endl<<"BASE    ==> "<< setw(5)<<base
           <<endl<<"HEIGHT ==> "<< setw(5)<<height
           << endl<< "AREA    ==> "<< setw(5)<<area;
      return 0;
}
```

عند تنفيذ هذا البرنامج ستظهر الرسالة الاولي على شاشة العرض لتعريف المنفذ بادخال قاعدة المثلث وليكن 15.5على النحو التالي :-

Please enter base of triangle ==>: 15.5

يلي ذلك ظهور الرسالة الثانية مطالبة باعطاء ارتفاع هذا المثلث وليكن 9.5 كالآتي:-

```
Please enter height of triangle ==>: 9.5
```

عندها يتم طباعة جميع البيانات المدخلة بما فيها مساحة المثلث المحسوبة كما يلي:-

```
BASE    ==> 15.5
HEIGHT ==> 9.5
AREA    ==> 73.625
```

(5.3 المؤثرات العلائقية Relational Operators

توجد ست مؤثرات علائقية يستطيع المبرمج استخدامها على أي زوج من العناصر يكون ناتجها القيمة خطأ (false) أو الرقم (0) أو القيمة صواب (true) أو الرقم (1) أو أي رقم آخر ماعدا الرقم (0)، وهذه المؤثرات هي:-

المؤثر	معناه
==	يساوى
!=	لا يساوى
<	اقل من
<=	اقل من أو يساوى
>	اكبر من
>=	اكبر من أو يساوى

مثال 3-12) خذ مثلا التعبير

'A'<'B'

هنا تتم المقارنة بين الحرف A وقيمته 65 في نظام (ASCII) مع الحرف B وقيمته 66 في نفس النظام أنظر ملحق (2) حيث ينتج عن هذه المقارنة

القيمة المنطقية (true) أو القيمة 1.

اما في حالة التعبير

```
'a'<'A'
```

فسوف يعطي النتيجة المنطقية (false) أو القيمة 0 والسبب هو أنه قيمة الحرف a في النظام هى 97 لا تكون اصغر من قيمة الحرف A وهي 65.

مثال 3-13) افرض أنه لدينا الاعلان التالي:-

```
char x='w';
int i=-3;
int j=5;
```

عليه فالجدول التالي يوضح مجموعة من التعبيرات المنطقية والنتائج المناظرة لكل تعبير:-

التعبير	القيمة
i !=j	1
j*2==i*2	0
'a' -1<x	1
i<j<2	1
!(5+j)	0
x=='a'	0
'x'==x+1	1
i-j<i+j	1
!x	0
i-j/2<=i*j/3	1
j<i>i+j	0

في التعبير (I < j < 2) تتم المقارنة بين i , j أولا وينتج عنه القيمة 1، ثم تقارن هذه القيمة مع العدد 2 فتكون النتيجة بطبيعة الحال القيمة 1، أما فيما يخص التعبير (i-j<i+j) فيعطي القيمة 0 والسبب ان 2 أكبر من 8-، كذلك بالنسبة للتعبير 'x'= =x+1 فهو يعني أضف 1 للقيمة المقابلة لقيمة x وهو الحرف w في نظام (ASCII) سينتج عنه الحرف الموالي للحرف w وهو الحرف x ، ومن ثم فأن التعبير يصبح كالتالي:-

x==x

والنتيجة في هذه الحالة ستعطي القيمة 1.

6.3) المؤثرات المنطقية Logical Operators

ايضا هناك عدد من المؤثرات المنطقية ينتج عنها اما قيمة 1 أي صحيح (true) أو القيمة 0 أي خطأ (false) وهذه المؤثرات هي:-

المؤثر	معناه
&&	مؤثر ثنائي AND ، يكون التعبير صحيحا اذا كان كل من العنصرين صحيحا.
||	مؤثر ثنائي OR ، يكون التعبير صحيحا اذا كان أحد العنصرين أو كليهما صحيحا .
!	مؤثر آحادي NOT ، نفي العنصر أوالنقيض

خذ مثلا التعبير

(a>0) && (a<=b)

يكون صحيحا true اذا كانت قيمة a أكبر من صفر وأقل من أو تساوي قيمة b، اما اذا كانت قيمة a أقل من صفر أو أكبر من قيمة b فأن ناتج التعبير سيكون خاطئا.

في حين ان التعبير

(2>7) || (5<6)

هنا يقارن التعبير الاول 2>7 وينتج عنه قيمة خطأ اي 0 ثم يقارن التعبير الثاني 6 > 5 وينتج عنه قيمة صواب أي 1، وبما ان التعبير الثاني كان صحيحا، عليه سيكون ناتج هذا التعبير القيمة العددية 1، في حين المؤثر(!) يعكس قيمة التعبير المنطقي، فمثلا

(! found)

يكون خاطئا اذا كانت قيمة found صحيحا.

7.3) مؤثرات الزيادة والنقصان Increment Decrement Operators

تتميز لغة سي++ ببعض المؤثرات التي قد لا توجد في بعض اللغات الاخرى وهي مؤثر الزياده ++ ومؤثر النقصان -- حيث يمكن استعمالهما مع المتغيرات فقط.

والشكل العام قد يأخذ الصورة التالية

op VariableName

أو الصورة الموالية

VariableName op

حيث:

op مؤثر الزيادة أو مؤثر النقصان.

VariableName متغير تم الاعلان عنه مسبقا.

يمكن وضع المؤثر ++ أو المؤثر -- في بداية المتغير أو في نهايته، ولكن يجب الاخذ في الاعتبار ان هناك فرقا كبيرا بين الوضعين.

فالتعبير count ++ يعني اضف العدد 1 لقيمة المتغير count قبل احتساب التعبير. على حين التعبير ++ count يعني استخدم القيمة الحالية للمتغير count ثم اضف العدد 1 الى قيمة المتغير.

أي أنه عند استعمال count++ نزيد قيمة count قبل احتساب التخصيص وعند استعمال ++count يحسب التخصيص قبل زيادة count، وينطبق هذا ايضا على حالة مؤثر النقصان (--).

مثال 3-14) ما هو ناتج البرنامج الآتي ؟

```
#include <iostream.h>
main()
{
	int i=2, j=2;
	int k=++i;
	cout<<"I="<<i<<"  K="<<k<<"\n";
	k=j++;
	cout<<"J="<<j<<"  K="<<k;
	return 0;
}
```

الناتج هو كالآتي:-

I=3 K=3

J=3 K=2

حيث تمت اضافة العدد 1 للمتغير i في التعبير ;k=++i فاصبحت قيمته تساوي 3 ثم خصصت هذه القيمة للمتغير k وبالتالي فهما يحتويان على نفس القيمة وهو العدد 3 وهذا ما طبع في السطر الاول.

اما في التعبير الثاني وهو ;++k=j هنا خصص العدد 2 وهي القيمة الاصلية للمتغير j الى المتغير k ثم اضيف العدد 1 الى المتغير j فاصبح يحتوي القيمة 3 وهي نتيجة طباعة السطر الثاني.

مثال 3-15) افرض أن المتغيرات c,b,a هي من النوع الصحيح وخصصت القيمة 3 لكل واحدة منها، فيما يلي مجموعة من المؤثرات مقرونة بالنتائج أمام كل منها:-

التعبير	القيمة
a++	3
++b	4
c--	3
- - -a	-3
- ++a	-4
a- --b	1
b++ + c++	6
++b + ++c	8
++b + c++	7
b-- + c--	6
++c+c	8
-a+ --b+ ++c	3
a++/2+ --b* ++c	9
--c* ++b- --a	6

8.3) المؤثرات المركبة Compound Operators

هناك ميزة أخرى تضاف الى لغة سي++ وهي أستخدام المؤثرات الحسابية المعروفة مثل (+ , - , * , / , %) والمؤثرات الخاصة بالبت (bit) التي سيأتي تناولها فيما بعد ان شاء الله ومنها (>> , << , & , |) مع اشارة التخصيص (=) تحت اسم يعرف بالمؤثرات المركبة ، وهي تعتبر طريقة مختصرة لجملة التخصيص.

الشكل العام

```
VariableName op=Expression
```

حيث op يمكن أن يكون احدى المؤثرات المذكورة اعلاه.

مثال 3-16) التعبير باستخدام جملة التخصيص المعروفة

```
x=x+9;
```

هو مكافي للتعبير المركب

```
x+=9;
```

وكلاهما يعني اضف 9 للمتغير القديم x ثم خصص هذه القيمة للمتغير الجديد x الموجود في الطرف الايسر.

مثال 3-17) أوجد ما الذى يطبعه البرنامج التالي ؟

```
#include <iostream.h>
main()
{
    int p=2, q=3, r=10;
    r/=p+q;
    cout<<"R==>"<<r;
    return 0;
}
```

الجواب: سيطبع الناتج المشابه للآتي:-

R==> 2

واذا ما غيرنا التعبير

r/=p+q;

الى الطريقة المعتادة وهي

r=r/p+q;

هنا ستكون نتيجة المتغير r هي القيمة 8 والسؤال هنا ما هو الاختلاف في الحالتين ؟

والجواب هو أن التعبير الاول ;r/=p+q يعني اجمع قيمتي المتغيرينq,p أولا ثم اقسم نتيجة الجمع على قيمة المتغير r أي ان هذا التعبير مكافيء للتعبير ;r=r/(p+g)

في حين أن التعبير الثاني ;r=r/p+q يعني اقسم قيمة المتغير r على قيمة المتغير p ، ثم اضف ناتج القسمة الى قيمة المتغير q واخيرا خصص ناتج هذه العمليات الى المتغير r فينتج عنها القيمة 8.

وعلى نفس الطريقة التعبير ;i*=j-k

مكافيء للتعبير i=i*(j-k)

وليس مكافئا للتعبير ;i=i*j-k

خذ مثلا آخر التعبير ;a%=b*c/c

مكافيء للتعبير ;a=a%(b*c/c)

9.3) مؤثر الفاصلة Comma Operator

مؤثر الفاصلة (,) من المؤثرات التي تتميز به لغة سى++ حيث له أدنى أفضلية بين كل المؤثرات في لغة سي++ وله الشكل التالي:-

exp1,exp2,...,expn;

حيث expn,..., exp2 , exp1 عبارة عن عملية تحتوي على تعبيرين أو أكثر.

وتكون التعابير مفصولة عن بعضها البعض بفاصلة، وبالتالي فان القيمة النهائية تحسب من التعبير الموجود من ناحية اليسار الى ناحية اليمين، ونوعها هو نوع التعبير الموجود بالطرف الايمن.

مثال 3-18) البرنامج التالي يبين استخدام مؤثر الفاصلة.

```
#include <iostream.h>
main()
{
        short j=2;
        short k=15;
        j+=5,k--,j+=k,k-=1;
        cout<<"J= "<<j<<"  K="<<k<<endl;
        short a=5;
        short b=4;
        b=a=3*b-1,b+=a;
        cout<<"While A="<<a<<"   B= "<<b;
        return 0;
}
```

في هذا البرنامج نجد مؤثر الفاصله بالامر الاول

j+=5,k--,j+=k,k-=1;

وبه 4 تعابير وهي :

1) التعبير الاول j+=5 وهو يعني اجمع القيمة 5 على قيمة المتغير j الابتدائية 2 وبالتالي تخصيص حاصل عملية الجمع للمتغير j.

2) التعبير الثاني --k وينتج عنه طرح 1 من قيمة المتغير k لتصبح قيمته 14.

3) التعبير الثالث j+=k وينتج عنه اضافة قيمة k الناتجة من التعبير الثاني وهي 14 الى القيمة القديمة للمتغير j وهي 7 واسنادها لنفس المتغير j لتصبح قيمته الجديدة تساوي 21.

4) التعبير الرابع والاخير k-=1 والذي ينتهي بالفاصلة المنقوطة (;) للدلالة على نهاية هذه الجملة يعني اطرح 1 من قيمة المتغير k لتصبح 13.

وكما نلاحظ في هذه التعابير الاربعة أنه تم تنفيذها من اليسار الى اليمين.

ثم يأتي مؤثر الفاصلة في الامر الثاني

```
b=a=3*b-1,b+=a;
```

حيث ينفذ التعبير

```
b=a=3*b-1,
```

ويتم التعويض في هذا التعبير بقيمة المتغير b القديمة وينتج عنه اسناد القيمة 11 الى المتغير a أولا ثم اسناد نفس القيمة للمتغير b بعد هذا يصل التنفيذ الى التعبير

```
b+=a;
```

وهو يعني اضافة قيمة المتغير a وهي 11 الى قيمة المتغير b لتصبح قيمته تساوي 22 عموما سوف يكون ناتج تنفيذ هذا البرنامج مشابها للآتي:-

```
J=21K=13
While A=11 B=22
```

10.3) اولويات تنفيذ المؤثرات Operators Precedence

تتضح الحاجة الماسة الى اولويات التنفيذ في حالة وجود تعبير به عدد من المؤثرات الحسابية والمنطقية المختلفه، واذا اردنا أن يأخذ التعبير مسارا محددا لتقييمه يجب استخدام الاقواس وذلك حسب ما يقتضيه التعبير، ويتم تنفيذ المؤثرات في لغة سي++ وفق سلم الاولويات حسب الترتيب التالي وذلك من أعلى الى أسفل حسب الجدول التالي:-

المؤثر	التنفيذ
()	من اليسار الى اليمين
! , -(unary) , ++ , --	من اليمين الى اليسار
* , / , %	من اليسار الى اليمين
+ , -	من اليسار الى اليمين
< , <= , > , >=	من اليسار الى اليمين
== , !=	من اليسار الى اليمين
&&	من اليسار الى اليمين
||	من اليسار الى اليمين
= , *= , /= , %= , += , -= , &= , ^= , |= , <<= , >>=	من اليمين الى اليسار
, (comma operator)	من اليسار الى اليمين

11.3) الدوال الرياضية Mathematical Functions

تزودنا لغة سي++ بمجموعة من الدوال الرياضية تعرف باسم الدوال القياسية (Standard Functions) كالموجودة في بعض الآلات الحاسبة الصغيرة،

تحت ملف العناوين <math.h> الذي يحتوي على تعريفات هذه الدوال، وفيما يلي بعض هذه الدوال مع امثلة عليها.

الدالة	الغرض منها	مثال	النتيجة
fabs(x)	القيمة المطلقة	fabs(-10)	10
sin(x)	الجيب	sin(90.0)	0.893997
cos(x)	جيب التمام	cos(90.0)	0.893997
tan(x)	الظل	tan(0.0)	0
asin(x)	معكوس الجيب	asin(1.0)	1.570796
acos(x)	معكوس جيب التمام	acos(1.0)	0
atan(x)	معكوس الظل	atan(1.0)	0.785398
sinh(x)	الجيب الزائد	sinh(0.0)	0
cosh(x)	جيب التمام الزائد	cosh(1.0)	1.543081
tanh(x)	الظل التعظيمي	tanh(1.0)	0.761594
exp(x)	القيمة الاسية	exp(3.0)	20.085537
log(x)	اللوغاريتم الطبيعية	log(20.085537)	3
log10(x)	اللوغاريتم العشرية	log10(100.0)	2
pow(x,y)	x بقوة y	pow(2,7)	128
		pow(9,0.5)	3
sqrt(x)	الجذر التربيعي	sqrt(16.0)	4
ceil(x)	تقريب قيمة x الى اصغر قيمة لا تكون اقل من x	ceil(6.8)	7
		ceil(-6.8)	-6

الدالة	الغرض منها	مثال	النتيجة
fmod(x,y)	باقي قسمة x/y	fmod(18.56,3)	0.56
		fmod(17.123,2.3)	1.023
floor(x)	اكبر قيمة صحيحة اقل من او تساوي x	floor(6.2)	6
		floor(-6.8)	-7
		floor(6.8)	6

يجب الأخذ في الاعتبار عند استخدام كل من الدوال sin و cos و tan أن تكون بالتقدير الدائري ويكون المتغير x من النوع المضاعف (double).

مثال 3-19) البرنامج الموالي يوضح كيفية استخدام واحدة من هذه الدوال الا وهي الدالة المشهورة pow .

```
#include <iostream.h>
#include <math.h>   // for pow
main()
{
	double x,y;
	cout<<"Enter value of x and y ==>: ";
	cin>>x>>y;
	cout<<x<<" power "<<y<<" = "<<pow(x,y);
	return 0;
}
```

وقت التنفيذ سيكون الناتج مشابها للآتي:-

```
Enter value of x and y ==>: 16  2
16 power 2 = 256
```

12.3) تمارين Exercises

(1) أي من جمل التخصيص الاتية مقبولة ؟

```
(a) -sum += value;          (b) ab=a+b**2;
(c) ave=x+y+z/5;            (d) m=n=p=10;
(e) f=++5 + --m;            (f) student_grade=test1 + final;
```

(2) أي من التعابير التالية تعطي نتيجة (true) صواب:-

```
(a) "test" == "tets"
(b) "ALL done"!= "all done"
(c) "a">="B"
```

(3) اكتب برنامجا لايجاد قيمة المعادلة التالية:-

$ansr=4xy^n+5z-10$

(4) بافتراض ان المتغيرات c,b,a من النوع الصحيح int مع تحديد القيم 9,10,11 لها على التوالي، المطلوب ايجاد ناتج الآتي:-

```
(a) a+=a;
    b+=b;
    c+=a++ - ++b;
    cout<<"A="<<a<<" B="<<b<<" C="<<c<<endl;
(b) a+=++a;
    b+=--b;
    c+=++c+ --c;
    cout<<"A="<<a<<" B="<<b<<" C="<<c<<endl;
(c) a=b=(a*b)-c;
    b-=--b+c++;
    c+=a++ + c--;
    cout<<"A="<<a<<" B="<<b<<" C="<<c<<endl;
```

(5) باستخدام الاعلانات والتخصيصات التالية:-

```
float a=5,b=3,c=8,d=2;
char x='w';
```

اوجد ناتج الطباعة للجمل التالية:-

(a) `cout<<(c-b>d*2)<<endl;`

(b) `cout<<(c==(a-4)*d)<<endl;`

(c) `cout<<(x-1!='u')<<endl;`

(d) `cout<<(!(!a+!b)==0)<<endl;`

(e) `cout<<((c/d>=4) && (c/d/d==d))<<endl;`

(f) `cout<<('y'==x+2)<<endl;`

(g) `cout<<((int(a*b*c)%2>0) || (!(c-a*d-2)==0))<<endl;`

(6) وضح اولويات تنفيذ مع بيان قيمة كل جملة من جمل التخصيص التالية:-

(a) `x=pow(sqrt(9.0),2);`

(b) `x=8+3*4/2-1;`

(c) `x=sqrt(pow(9.0,2))+3/2+1;`

(d) `x=4%4+4*4/4;`

(e) `x=sqrt(pow(-9.0,2))-9 %5;`

(f) `x=7%(5%3)`

(g) `x=7%5%3;`

(h) `x=(3*9*(3+(9*3/3))*3)-3;`

(i) `x=ceil(8.0/5.0)+8>2*4;`

(j) `x=fmod(34.66,2.3)+floor(-7.8);`

(7) اكتب التعابير الآتية باستخدام الاقواس تبعا لاولوية التنفيذ .

(a) `x+y/z*w`

(b) `a+5*c%d-c`

(c) `x=y=z=55`

(d) `r==x && y !=10`

(e) `m && n && x || y || z`

(f) `a & !b || c && d !e`

(8) اكتب برنامجا مهمته مطالبة المستخدم بادخال قيمتين مع ايجاد مجموعهما وحاصل ضربهما ومتوسطهما بحيث تكون صورة شاشة العرض في حالة ادخال القيمتين 10,15 على النحو التالي:-

```
Input two numbers ==>: 10  15
sum is 25
product is 150
average is 12.5
```

(9) تتبع واستنتج مخرجات البرنامج التالي:-

```
#include<iostream.h>
main()
{
       int a=4,b=3,c=5;
       a+=a++;
       b=--a+c;
       c=++a + b++;
       cout<<"A="<<a<<"  B="<<b<<"  C="<<c<<endl;
       b=10;
       c=2;
       a+=b=b/c;
       c=b++ + --a;
       cout<<"A="<<a<<"  B="<<b<<"  C="<<c<<endl;
       b+=c+=a-(a%b);
       a+=c=(b+c);
       cout<<"A="<<a<<"  B="<<b<<"  C="<<c<<endl;
       return 0;
}
```

(10) اكتب برنامجا مهمته حساب مساحة area وحجم volume الكرة باستخدام نصف القطر r حيث

$area=4\pi r^3$

$volume=\frac{4\pi r^2}{3}$

(11) قم بكتابة برنامج يقوم بقراءة اسم الطالب ودرجاته في ثلاثة امتحانات بالشكل التالي :-

Type student name: aaaa bbbb

Type first test : 70

Type second test : 80

Type third test : 60

ثم قم بطباعة متوسط هذه الدرجات مع البيانات السابقة بالشكل التالي :-

Student name	test-1	test-2	test-3	average
Aaaa bbbb	70.0	80.0	60.0	70.0

(12) أكتب برنامج عند تنفيذه ينتج عنه جدول الصدق (truth table) باستخدام متغيرات Q, P على أن يكون الجدول بالشكل التالي :-

Truth table for P and Q

P	Q	P&&Q	P||Q	!P
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

الفصل الرابع: جمل الاختيارات

تناولنا في الفصول السابقة بعض البرامج التي تتكون من تعليمات يتم تنفيذها بصورة متتابعة ودون أي اختيارات، حيث لا يمكن القفز وتخطي تعليمات معينه في البرنامج الواحد، وحيث أنه من الضروري في كثير من الاحيان تحويل مسار البرنامج من مكان الى مكان آخر، عليه فسوف نتناول في هذا الفصل مجموعة من جمل الاختيار وكيفية استخدامها حتي تحقق المطلوب منها.

1.4) جملة اذا The if Statement

من وظائف هذه الجملة تنفيذ جملة أو عدد من الجمل عندما يتم تحقيق تقويم التعبير المنطقي التابع لها والذي يحدد نتيجة القرار أما صحيح (true) واما (false) وهو ما يسمى هنا بالاختيار ذي التفريعين (two way hoices)، وتاخذ جملة if الشكل التالي:-

```
if (Logical_Expression)
      Yes_Statement
next statement
```

يتم تنفيذ جملة Yes_Statement اذا كانت نتيجة الشرط Logical_Expression والذي يجب وضعه ضمن القوسين () صحيحة، اما اذا كانت نتيجته خاطئة فعندها يتم تنفيذ الجملة التالية next statement مع ملاحظة أن الجملة التالية تنفذ سواء كان الشرط صحيحا أو خاطئا.

مثال 4-1) انظر الى هذا الايعاز

```
if( grade >= 50 )
    cout<< "PASS "<<grade;
```

هنا اذا كانت قيمة المتغير grade أكبر من 50 أو مساوية لها عندها يتم طباعة كلمة PASS مع قيمة grade والا فلن تتم هذه الطباعة .

مثال 4-2) خذ مثلا الجمل التالية:-

```
if (grade >=85)
    cout<< "Congratulations !";
cout<<"Your Grade is "<<grade;
```

هنا اذا تحقق الشرط وهو (grade >=85) عندها تطبع كلمة التهنئة وهي

Congratulations !

بعدها تنفذ الجملة التالية، وهي طباعة الدرجة grade، اما اذا لم يتحقق الشرط، فعندها لا تطبع التهنئة بل تطبع الدرجة فقط.

مثال 4-3) ما هو ناتج تنفيذ البرنامج الآتي:-

```
#include<iostream.h>
main()
{
    char ch;
    cout << "\nEnter the character ==>: " ;
    cin >> ch ;
    if( (ch>='a') && (ch<='z') )
    cout<<"The character you type [" << ch<< "] is small character";
    cout<<"\nAll done";
    return 0;
}
```

الجواب: في حالة ما اذا تم ادخال القيمة a للمتغير ch على النحو التالي :-

Enter the character ==>: a

حينها سيطبع البرنامج السطرين التاليين :-

The character you type [a] is small character

All done

لان الشرط (ch>='a') && (ch<='z') تحقق وبالتالي تم تنفيذ جملة الطباعة الموالية لـ if يتبع ذلك جملة الطباعة التالية التي بها الرسالة All done.

أما في حالة ادخال القيمة الحرفية F الى نفس المتغير ، فالشرط لجملة if هنا لم يتحقق وبالتالي تم تجاهل جملة الطباعة الاولى ونفذت جملة الطباعة الثانية التي ينتج عنها الرسالة

All done

2.4) الجملة المركبة Compound Statement

الجملة المركبة تعني انها تحتوي على تسلسل من جملتين متتاليتين أو اكثر محاطة بقوسي الفئه { }، ويستخدم هذا النوع من الجمل في حل المسائل التي تتطلب تنفيذ مجموعة من الاوامر وذلك تحت شرط معين سواء باستخدام جمل الاختيار أو جمل التكرار وغيرها.

خذ مثلا جملة if التالية:-

```
if(Logical_Expression)
{
      Yes_Statement_1;
      Yes_Statement _2;
      . . .
      Yes_Statement _Last;
}
```

هنا اذا تحقق الشرط Logical_Expression عندها تنفذ الجملة المركبة أي كل الجمل المحصورة بين قوسي الفئة { }، أما اذا لم يتحقق الشرط فانه سيتم تجاهل هذه الجمل جميعها وينفذ الجملة التي تلي القوسين { }.

فيما يلي بعض الامثلة التي توضح أستخدام الجملة المركبة

(1)

```
if((number <50) || (counter !=10))
{
     counter +=1;
     sum +=number;
}
```

(2)

```
if(ch !='?')
{
     cin>>letter;
     cout<< letter;
}
```

مثال 4-4) المطلوب كتابة برنامج مهمته قراءة متغيرين من النوع الحقيقي مع طباعتهما تصاعديا.

```
#include<iostream.h>
main()
{
      float x,y, temp;
      cout<<" Enter 2 float numbers ==>: ";
      cin>>x>>y;
      if(x>=y)
      {
           temp=x;    // Store old x in temp
           x=y;       //  Store old y in x
           y=temp;    //  Store old temp in y
      }
      cout<<"X="<<x<<"  Y="<<y;
      return 0;
}
```

وقت تنفيذ البرنامج وبعد ظهور الرسالة على الشاشة يتم ادخال القيمتين

```
Enter 2 float numbers ==> :  4.5    9.5
```

هنا يتم تخصيص القيمتين 4.5 , 9.5 للمتغيرين x , y على التوالي، وبالتالي فالشرط (x>=y) لم يتحقق والسبب أن قيمة المتغير x وهي 4.5 لم تكن أكبر أو تساوي قيمة المتغير y وهي القيمة 9.5 لذلك لم تنفذ الجمل المحاطة بالقوسين { }والموالية لجملة if وتبقي القيمتان بدون تبديل وينتقل التنفيذ الى جملة الطباعة مباشرة وينتج عنها السطر المشابه للآتي:-

```
X=4.5   Y=9.5
```

اما اذا ما نفذ هذا البرنامج مرة أخرى وادخلت القيمتان بالشكل التالي:-

```
Enter 2 float numbers ==> :9.5   4.5
```

وبالتالي تسند القيمة 9.5 للمتغير x والقيمة 4.5 للمتغير y يأتي بعدها عملية المقارنة والتي ينتج عنها تحقق الشرط (x>=y) وتنفيذ الجملة المركبة:

```
{
      temp=x;
      x=y;
      y=temp;
}
```

والتي مهمتها تبديل قيمة المتغيرين عن طريق متغير ثالث temp ومن ثم طباعتهما تصاعديا كما يلي:-

```
X=4.5   Y=9.5
```

3.4) مؤثر النطاق Scope Operator

تسمح لغة سي++ بالاعلان على متغير عام global ومتغير محلي local بنفس الاسم باستخدام المؤثر (::) الذي يتخطي أي عدد من الجمل المركبة المتداخلة والوصول الى المتغير الخارجي واستعماله في أي نقطة من البرنامج.

مثال 4-5) البرنامج التالي تم فيه الاعلان عن متغيرين بالاسم number ، واحد في النطاق العام والآخر في النطاق المحلي.

```
// Using the scope operator
#include<iostream.h>
double number=12.345;          // global variable
main()
{
        int number=1000;           // local variable
        cout<<"The local value = "<<number<<endl;
        cout<<"While global value = "<< :: number<<endl;
        return 0;
}
```

في جملة الطباعة الاولي بهذا البرنامج سوف تطبع قيمة المتغير المحلي وهي القيمة 1000، بينما استخدم المؤثر (::) مع جملة الطباعة الثانية وسبب ذلك الوصول الى قيمة 12.345 والمسندة للمتغير العام الذي هو خارج نطاق الدالة الرئيسية () main، عموما التنفيذ يعطي الناتج الموالي

```
The local value = 1000
While global value = 12.345
```

ايضا تسمح لنا لغة سي++ باستخدام ميزة أخرى وهي الاعلان عن متغيرين داخل الدالة الرئيسية بنفس الاسم في اكثر من جملة مركبة حتي وان

كانت هذه الجمل المركبة متداخلة، حيث يكون المتغير معروفا في نطاق الجملة المعلن فيها.

مثال 4-6) البرنامج الموالي يوضح استخدام هذه الطريقة وما يترتب عليها.

```
#include<iostream.h>
main()
{
        int m=1000;
        {
              double m=12.345;
              cout<<"inside block M= "<<m<<endl;
        }
        cout<<"outside block M= "<<m<<endl;
        return 0;
}
```

هنا تم الاعلان عن المتغير m من النوع الصحيح وخصصت له القيمة 1000 في بداية البرنامج ، وبالتالي فهو يعتبر خارجياً بالنسبة للقالب (Block) الذي يضم جملتين ، الاولي الاعلان عن المتغير m من النوع المضاعف وخصصت له القيمة 12.345، الثانية طباعة قيمة هذا المتغير.

وفي حالة تنفيذ هذا البرنامج سينتج عن الطباعة الاولي القيمة 12.345 في حين جملة الطباعة الثانية سينتج عنها طباعة القيمة 1000، وهذا هو الناتج.

```
inside block M=12.345
outside block M=1000
```

4.4) جملة اذا – فإن – وإلا The if-else Statement

تسمي هذه الجملة بالجملة الكاملة وتعني انه اذا تحقق الشرط فافعل كذا والا فافعل كذا.

الصيغة العامة

```
if(Logical_Expression)
    Yes_Statement;
else
    No_Statement;
Next_Statement;
```

هنا لو تحقق الشرط Logical_Expression عندها نفذ الجملة Yes_Statementواذا لم يتحقق الشرط فنفذ الجملة No_Statement اما فيما يخص الجملة Next_Statement فهي تنفذ في كلا الحالتين.

فالايعاز

```
if (grade >=50)
   cout<< "PASS"<<grade;
else
   cout<<"FAIL"<<grade;
```

يتكون من ثلاثة عبارات الاولى الشرط (grade>=50) والثانية العملية التي يجب تنفيذها اذا تحقق الشرط وهي طباعة كلمة PASS مع قيمة الدرجة grade، والثالثة العملية التي يجب تنفيذها اذا لم يتحقق الشرط وهي طباعة كلمة FAIL مع قيمة الدرجة grade.

ايضا قد تستخدم الجملة المركبة مع جملة if ، فالامر

```
if(circle)
{
   cin>>radius;
   area=3.14159 * redius * redius;
   cout<<"Area of circle"<<area;
}
else
{
   cin>>length<<width;
   area=length * width;
   cout<<"Area of rectangle"<<area;
}
```

يعني اذا تحقق شرط جملة if عندها يتم تنفيذ الجملة المركبة التي تحتوي على ثلاثة اوامر اي ادخال قطر الدائرة redius ومن ثم حساب وطباعة مساحتها area ، اما اذا لم يتحقق الشرط فحينها يتم ادخال طول المستطيل length وعرضه width وبالتالي حساب وطباعة مساحته area .

5.4) المؤثر الشرطي The Conditional Operator

ميزة اخري تضاف الى لغة سى++ وهي وجود المؤثر الشرطي والذي يرمز له بعلامة الاستفهام (?) وعادة ما يستخدم في المقارنة بين قيمتين وله الصيغة العامة التالية :-

exp1 ? exp2 : exp3;

ويتكون مؤثر (?) من ثلاثة تعبيرات حيث يقوم بتقييم التعبير الاول exp1 اولا على النحو التالي:-

اذا كانت قيمته صحيحة لا تساوي صفرا ، عندها تحسب قيمة التعبير الثاني exp2 وبالتالي تكون هذه القيمة هي قيمة التعبير النهائية.

أما اذا كانت قيمة التعبير الاول exp1 خاطئه أي تساوي صفرا فتحسب قيمة التعبير الثالث exp3 ، وتكون هذه القيمة هي قيمة التعبير النهائية.

مثال 7-4) البرنامج التالي يجرى توضيح استخدام المؤثر الشرطي.

```
#include<iostream.h>
main()
{
	float grade;
	cout<<"Type the grade ==>:";
	cin>>grade;
	grade>=50 ? cout<<"passed"<<endl : cout<<"failed"<<endl;
	return 0;
}
```

في هذا البرنامج يتم ادخال الدرجة grade ولتكن 80 أولا كما يلي:-

Type the grade ==>:80

يأتي بعدها تقييم التعبير الاول الذي يمثله الشرط grade>=50 وحيث ان قيمة المتغير grade اكبر من 50 عليه تكون نتيجة هذا الشرط صحيحة وبالتالي تطبع كلمة passed ، اما اذا تم ادخال الدرجة 35 ، هنا الشرط grade>=50 لم يتحقق وبالتالي تطبع failed.

مثال 4-8) الجزء التالي من البرنامج

```
if(x>=0)
  s=x;
else
  s=-x;
```

يقابله الامر التالي:-

```
x>=0 ? s=x : s=-x;
```

مثال 4-9) المطلوب كتابة برنامج كامل مهمته قراءة N من القيم الصحيحة مع ايجاد وطباعة عدد القيم الفردية والزوجية ومجموع كل واحدة منها وذلك باستخدام مؤثر الشرط (؟).

```
#include<iostream.h>
main()
{
      int i,n,odd,sum_odd,even,sum_even,value;
      even=odd=sum_odd=sum_even=0;
      cout<<"Type number of values ==>:" ;
      cin >>n;
      cout<<"Please enter "<<n<<" values ==>:" ;
      for(i=1;i<=n;i++)
      {
          cin>>value;
```

```
        value %2!=0 ? odd++,sum_odd+=value :
              (even++,sum_even+=value);
    }
    cout <<"We have "<< odd<<" odd numbers"
        <<"and their sum is "<< sum_odd<<endl;
    cout <<"We have " << even<<" even numbers"
        <<"and their sum is "<< sum_even<<endl;
    return 0;
}
```

في هذا البرنامج استخدمت جملة for لتكرار مجموعة من الجمل n من المرات (والتي سوف يأتي شرحها في الفصل القادم ان شاء الله)، وبداخلها التعبير الاول من مؤثر الشرط وهو الشرط value % 2!=0 والذي يعني اذا كان ناتج بقية قسمة المتغير value على القيمة 2 لا يساوي 0 عليه تكون القيمة المدخلة للمتغير value هي قيمة فردية ، وبالتالي ينفذ التعبير الثاني الذي يضم مؤثر الفاصلة وبه:-

التعبير الاول odd++ يستعمل كعداد للقيم الفردية .

التعبير الثاني sum_odd+=value ويستعمل لحفظ مجموع القيم الفردية.

أما اذا كان التعبير الاول وهو الشرط value % 2!=0 غير صحيح حينها تنفذ الجمل المحصورة بين الاقواس وهي:-

```
(even++,sum_even+=value)
```

ليحصي عدد القيم الزوجية ومجموعها.

وهذا هو الناتج اذا ما نفذ هذا البرنامج.

```
Type number of values ==> : 10
Please enter 10 values: 2  7  10  12  6  11  8  12  5  10
We have 3 odd numbers .
and their sum is 23
We have 7 even numbers .
and their sum is 60
```

6.4) اختبار قناة الادخال Testing Input Stream cin

توجد في لغة سي++ امكانية اختبار قناة الادخال لمعرفة ما اذا كانت تشتغل بالطريقة الصحيحة أم يوجد بها عطل، وفي الحالة الاخيرة تمكننا هذه اللغة من اصلاح هذا العطل.

مثال 4-10) الجملة

```
if (cin)
```

ترجع عن طريق دالة اختبار القناة وهي rdsate() بقيمة صفرية اذا كانت القناة cin عامله، أي انها على ما يرام وبقيمة 2 أو 3 اذا كانت القناة بها عطل، مع امكانية تصليح العطل اذا كان قابلا للاصلاح باستخدام الدالة cin.clear().

اما الجملة

```
if (cin>>num)
```

فتعني استقبل قيمة المتغير num ثم اختبر حالة القناة، فاذا كانت عامله نفذ الجملة الموالية.

مثال 4-11) البرنامج التالي يتم فيه استقبال متغير من النوع الصحيح ومن ثم اختبار قناة الادخال بقدرتها على اصدار الرسالة المناسبة في حالة ما تكون عامله أو باصدار رسالة أخرى مع اخراج نوع الحالة بعد الاصلاح.

```
#include <iostream.h>
#include <iomanip.h>
main()
{
        int mynumb;
        cout<< "Enter your number ==> ";
        if (cin>>mynumb)
            cout <<"The "<<mynumb<< " is an integer number, "
```

```
              <<"The state is "<<cin.rdstate()<<endl;
    else
    {
        cout<<"This is not an integer number "<<endl;
        cout<<"The state before repair is "<<cin.rdstate()<<endl;
        cin.clear();
        cout<<"The state after repair is "<< cin.rdstate();
    }
    return 0;
}
```

اذا ما نفذ هذا البرنامج وادخل الرقم 567 كما يلي

Enter your number ==>567

عندها يتم فحص القيمة المدخلة، وحيث انها قيمة مناسبة، فسوف تطبع الرسالة الموالية لجملة if وهي

The 567 is an integer number, The state is 0

اما في حالة ادخال القيمة الحرفية A مثلا فعندها يكون الرد هو:

This is not an integer number

The state before repair is 2

The state after repair is 0

7.4) جملة اذا المتداخلة The Nested if Statement

وهي جملة ذات علاقة شرطية وتتألف من مجموعة من جملة if مع أي عدد من else-if عليه يجب الانتباه عند استخدام هذا النوع من التداخل بدقة، حتى لا يكون هناك أية اخطاء، وهي قد تأخذ الشكل التالي:-

```
if(condition1)
        statement11;
else
        if(condition2)
        {
```

```
                statement21;
                statement22;
           }
           else
                if (condition3)
                     statement31;
```

وهي تعني اذا كان الشرط condition صحيحا تنفذ الجملة أو مجموعة الجمل الواقعة ما بين if وأول else if ويتم تجاهل بقية الجمل ، أما اذا كان الشرط خطأ فإنه يتم الانتقال الى أول شرط else if وهكذا.

ومن ناحية تنظيمية وحتي يكون هذا النوع من الجمل سهل المتابعة والفهم ولمنع الارتباك ، ينبغي أن تأتي كل كلمة else تحت كلمة if التي تتبعها كما هو موضح في الشكل العام.

مثال 4-12) المطلوب كتابة برنامج لقراءة قيمة حرفية بحيث يطبع الرسالة المناسبة التي تدل على نوعية الحرف الذي تم ادخاله هل هو حرف كبير أم صغير أم رمزا خاصا.

```
#include<iostream.h>
main()
{
     char ch;
     cout << "\Enter the charcter ==>: ";
     cin  >> ch;
     if( (ch>='a') && (ch<='z') )
          cout <<"The character you type [" <<ch
               <<"] is small character"<<endl;
     else
          if ((ch>='A') && (ch<='Z') )
               cout <<"The character you type ["<<ch
                    <<"] is capatial character"<<endl;

          if ((ch>='0') && (ch<='9') )
               cout <<"The character you type ["<<ch
                    <<"] is digit character"<<endl;
```

```
        else
                cout <<"The character you type ["<<ch
                     <<"] is special character"<<endl;
        cout<<"All done";
        return 0;
}
```

ناتج التنفيذ في حالة ادخال الحرف a سيكون:-

```
Enter the charcter ==>: a
The character you type [a] is small character
All done
```

عند كتابة هذا البرنامج، تم استخدام جملة if أكثر من مرة وهو ما يعرف بجملة اذا المتداخلة، حيث تمت مقارنة الحرف المدخل وهو المتغير ch مع مجموعة الحروف الصغيرة اولا، فاذا تحقق الشرط تطبع الرسالة الدالة على ذلك واذا لم يتحقق تأتي جملة if الثانية التي تلي else ويقارن المتغير ch مع الحروف الكبيرة وبالتالي اصدار الرسالة المناسبة، وعن طريق شرط جملة if الاخيرة يتم طباعة الرسالة التي تدل على ان الحرف المدخل هو من النوع الرقمي في حالة تحقق الشرط، والا فأن التحكم ينتقل الى الجملة التي تلي else الأخيرة أي ان الحرف المدخل هو حرف من النوع الرمزي الخاص.

لاحظ ايضا كيفية تنظيم جملة if-else المتداخلة التي اصبح من السهل بواسطتها فهم ومتابعة ما الذي يفعله هذا البرنامج.

مثال 4-13). البرنامج التالي يطبع اسم اللون حسب الحرف الاول من الكلمة الدالة على ذلك اللون سواء كان الحرف صغيرا أو كبيرا.

```
#include<iostream.h>
main()
{
	char color;
	cout<<"Give the color now ==>: ";
	cin>> color;
	if ((color =='r')||(color == 'R'))
		cout<<"Your color is RED" << endl;
	else
		if ((color =='o')||(color =='O'))
			cout<<"Your color is ORANGE"<<endl;
		else
			if((color == 'b')||(color =='B'))
				cout<<"Your color is BLUE"<<endl;
			else
				if( (color =='g')||(color=='G') )
					cout<<"Your color is GREEN"<<endl;
				else
					cout <<"Your selection [" << color
						<<"] out of range"<<endl;
	return 0;
}
```

استخدم في هذا البرنامج ادخال قيمة متغير من النوع الحرفي الذي يمثل الحرف الاول من بعض الالوان، تلا ذلك استعمال الكثير من جمل if المتداخلة للبحث عن اللون الذي يمثله الحرف المدخل، فمثلا الشرط

(color =='r')||(color == 'R')

يعني مقارنة المتغير color مع الحرف R أو الحرف r الذي تبدأ به كلمة اللون الاحمر RED فاذا ما تحقق هذا الشرط، عندها تطبع الرسالة المناسبة، واذا لم يتحقق هذا الشرط ينتقل التحكم الى الشرط التالي، وهكذا فاذا لم يتحقق أي شرط من الشروط الموجودة في هذا البرنامج، يجري تنفذ جملة الاخراج الاخيرة والتي تدل على ان اللون المطلوب خارج نطاق هذا البرنامج.

مثال 4-14) المطلوب كتابة برنامج لقراءة أي رمز مع عمل الآتي:-

1) اذا كان الرمز من النوع الحرفي أو الرقمي فانه يحل محله الرمز الموالي له فمثلا الحرف X يحل محله Y والرقم 5 يحل محله 6 فيما عدا:-

a) اذا كان الرمز رقم 9 يحل محله الرقم 0

b) اذا كان الرمز حرف z يحل محله الحرف a

c) اذا كان الركز حرف Z يحل محله الحرف A

2) حلول نجمة (*) محل أي رمز غير الحروف والارقام

3) عندما يتم ادخال الرمز (?) اطبع الرسالة المناسبة.

```
#include<iostream.h>
main()
{
    char char_in;
    cout<<"\nEnter a character ==>: ";
    cin>>char_in;
    if (char_in =='?')
        cout<<"Good bye no next character";
    else
    {
        char char_out;
        if ((char_in>='0') && (char_in<'9')
            ||(char_in>='a') && (char_in<'z')
            ||(char_in>='A') && (char_in<'Z') )
            char_out=char_in+1;
        else
            if (char_in=='9')
                char_out='0';
            else
                if (char_in=='z')
                    char_out='a';
                else
                    if (char_in=='Z')
                        char_out='A';
                    else
```

```
                        char_out='*';
    cout <<"The input character is ["
         <<char_in<<"]";
    cout <<" The next character is ["
         <<char_out<<"]";
    }
    return 0;
}
```

عند كتابة هذ البرنامج ونظرا لكثرة المقارنات ومن أجل الوصول الى المطلوب بالصورة المناسبة، تم استخدام عدد من جمل if المتداخلة، فالجملة الاولي الغرض منها الخروج من البرنامج بعد كتابة الرسالة المناسبة وذلك في حالة ادخال الرمز (?)، اما جملة if الثانية والذي يضم شرطها التعبير المنطقي

(char_in>='0') && (char_in<'9')

فهي لتحديد ما اذا كان الرمز المدخل من النوع الرقمي أم لا، بينما التعبيران المنطقيان

(char_in>='a') && (char_in<'z') || (char_in>='A') && (char_in<'Z')

لمعرفة ما اذا كان الرمز المدخل هو من النوع الحرفي الصغير أم الكبير، وهكذا نلاحظ أنه لكل جملة غرض معين.

وعموما فإن تنفيذ هذا البرنامج سيعطي الناتج التالي:-

```
Enter a character ==>: z
The input character is [z] and next character is [a]
```

لاحظ أن الجملتين التاليتين في البرنامج وهما:-

```
cout<<"The input character is ["<<char_in<<"]";
cout<<" The next character is ["<<char_out<<"]";
```

مهمتهما اخراج الرمز الذي تم ادخاله والموالي له ويتم تنفيذهما في كل مرة لا يكون فيها الحرف المدخل هو الرمز (?).

8.4) جملة التحويل The switch Statement

كما لاحظنا سابقا في جملة if أن المقارنة تتم بين قيمتين حيث تكون النتيجة إما صحيحة أو خاطئة ولكن في بعض الاحيان فإن حل المسألة يفرض على المبرمج المقارنة بين عدد من القيم تبعا لشروط مختلفة.

ولحل هذه المعضلة ، وعوضا عن استخدام جملة if يوجد امر التبديل أو التحويل switch الذي يقوم بعدة تحويلات مختلفة .

الشكل العام لجملة التحويل switch هو :-

```
switch (expression)
{
     case Constant_1 :  Statement_Sequence_1;
                        break;
     case Constant_2 :  Statement_Sequence_2;
                        break;
           . . .
     case Constant_n :  Statement_Sequence_n;
                        break;
     default : Default_Statement_Sequence;
}
```

حيث :-

switch كلمة محجوزة ومهمتها تحويل وتنفيذ واحد من عدة اختيارات متاحة وفقاً لقيمة التعبير expression وتبدأ switch وتنتهي بالقوسين { }.

expression هو التعبير الذي تكون نتيجته إما من النـوع الحرفـيchar أو النـوع الصـحيح int أو النـوع المنطقـي ((true) القيمـة 1) أو ((false) القيمة 0).

case تمثل القيمة المناسبة بعد احتساب التعبير expression.

Constant_1 ,..., Constant_n قيمة التعبير expression ويمكن أن يكون عددا من النوع الصحيح أو النوع الحرفي أو المنطقي، على أن يتبع هذه القيمة الشارحة (:) ولا تحتاج قيمة constant الناتجة من التعبير expression ان تكون في ترتيب معين.

break تستعمل عند آخر كل مجموعة جمل ضمن switch لتفادي اختيار بقية الحالات cases واذا لم تستخدم بعد أي حالة فان التنفيذ ينتقل الى الحالة الموالية لهذه الحالة مباشرة.

default وهي كلمة محجوزة وتعني حالة اسقاط وهي اختيارية وتنفذ عندما تكون قيمة التعبير expression لا تتحقق مع أي قيمة من قيم value.

مثال 4-15) انظر الى جمل if التالية :-

```
if(temp =1)
     cout<< " hot day ";
else
     if(temp=2)
        cout<< " worm day ";
     else
        if(temp=3)
           cout<< " cold day ";
        else
           cout<< "temp out of range ";
```

فيمكن استبدالها باستخدام جملة switch على النحو التالي :-

```
switch(temp)
{
     case 1 : cout<< "hot day ";
              break;
     case 2 : cout<< " worm day ";
              break;
     case 3 : cout<< " cold day ";
              break;
     default: cout<< "temp out of range";
}
```

مثال 16-4) اكتب برنامجاً لقراءة متغير من النوع الحرفي يمثل الحرف الاول من كلمة أي مؤثر حسابي ومتغيرين آخرين من النوع الحقيقي ، ثم اوجد نتيجة العملية الحسابية التي يدل عليها الحرف المدخل ، فمثلا عند ادخال الحرف M فهو يعني ضرب القيمتين ، بينما الحرف S يعني طرحهما، وهكذا، مع اظهار الرسالة المناسبة في حالة أن الحرف المدخل ليس من المؤثرات الحسابية.

```
#include <iostream.h>
main()
{
       float num1,num2,result;
       char op;
       cout<<"\nEnter character and 2 numbers ==>: ";
       cin>>op>>num1>>num2;
       switch(op)
       {
            case 'A' :
            case 'a' :   result=num1+num2;
                         cout<<num1<<" + "<<num2<<"="<<result<<endl;
                         break;
            case 'S' :
            case 's' :   result=num1-num2;
                         cout<<num1<<" - "<<num2<<"="<<result<<endl;
```

```
                break;
        case 'M' :
        case 'm' :  result=num1*num2;
                    cout<<num1<<" * "<<num2<<"="<<result;
                    break;
        case 'D' :
        case 'd' :  if(num2 ==0)
                      cout<<"***** Hi the input was error *****"<<endl;
                    else
                    {
                        result=num1/num2;
                         cout<<num1<<" / "<<num2<<"="<<result<<endl;
                    }
                    break;
        default :   cout<<"Sorry data error"<<endl;
    }
    return 0;
}
```

عند تنفيذ البرنامج تظهر الرسالة

```
Enter character and 2 numbers ==>:
```

طالبة ادخال حرف مع قيمتين ، يأتي بعدها جملة switch وهي لا تنتهي بفاصلة منقوطة (;) حيث تحتوي على التعبير op والذي هو من النوع الحرفي char فاذا ما تم ادخال الحرف A مع القيمتين 10 ,30 مثلا كالآتي:-

```
Type character and 2 numbers ==>: A 10 30
```

عندها يتم تنفيذ الحالة الاولي وهي:

```
case 'A' :
case 'a' : result=num1+num2;
          cout<<num1<<" + "<<num2<<"="<<result<<endl;
           break;
```

وتعني اجمع قيمة المتغيرين num1 , num2 وتخصيص الناتج للمتغير result ومن تم تنفيذ جملة الطباعة التي ينتج عنها السطر التالي:-

10 + 30=40

ثم ينفذ الامر break الذي بدوره يحول التحكم إلى الجملة التي تلي قوس الفئة المغلق } التابع لجملة switch وفي حالة عدم استخدام break في نطاق هذه الحالة سوف ينتقل التحكم إلى الحالة التالية مباشرة وهي:

case ‘S’:

وهذا غير صحيح، وذلك في حالة ادخال الحرف الكبير 'A' أو الحرف الصغير 'a' وهكذا يتم مع باقي الحالات، واذا تم تنفيذ هذا البرنامج لعدة مرات سينتقل التحكم الى الحالة المناسبة، أما في حالة ادخال اي حرف آخر غير الحروف المشار اليها، عندها تنفذ جملة الاسقاط default حيث يتم طبع الرسالة المناسبة وهي:

Sorry Data Error

مثال 4-17) البرنامج الموالي مهمته استقبال ثلاثة قيم لمتغيرات من النوع الصحيح مع ايجاد اكبر هذه القيم باستخدام جملة switch.

```
#include<iostream.h>
main()
{
        int x,y,z,w;
        cout<<" \nEnter value of x,y,z ==>: ";
        cin>>x>>y>>z;    // input 3 values
        switch (x>y)
        {
            case 1 :    switch(x>z)
                        {
                            case 1 : w=x; break;
                            case 0 : w=z; break;
```

```
                    }
                    break;
        case 0 :    switch(y>z)
                    {
                        case 1 : w=y;
                                 break;
                        case 0 : w=z;
                                 break;
                    }
    }
    cout<<"The larger value is "<<w;
    return 0;
}
```

كما نلاحظ في هذا البرنامج انه تم استخدام جملة switch المتداخلة، فالتعبير x>y استخدم مع جملة switch الاولى (الخارجية) فإذا كان هذا التعبير صحيحا أي قيمته 1 عندها ينتقل التحكم الى جملة switch الثانية (الداخلية) والذي استخدم فيها التعبير x>z ، فاذا كانت صحيحة فسوف تخصص القيمة الكبرى للمتغير w، وفي حالة ان التعبير x>y خطأ أي له القيمة 0، عندها تتم مقارنة التعبير y>z في جملة switch الثالثة فإذا كان صحيحا تخصص قيمة y للمتغير w والا فتخصص قيمة z للمتغير w، وفي حالة تنفيذ هذا البرنامج وادخال البيانات كالآتي:-

```
Enter value of x,y,z ==>: 20 30 10
```

سيكون الناتج السطر الموالي:-

```
The larger value is 30
```

9.4) تمارين Exercises

(1) اذا اعطيت التعريفات وجمل الاسناد التالية :-

```
int i=2,j=3,k=13;
float x=5.4;
char a='a',b='b'
```

فما هو ناتج الفقرات التالية :-

```
(a)  cout<<((i*j !=k) ? x:k);
(b)  cout<<((j%i>j) ? a:b);
(c)  cout<<((a==97) ? x*j:j*k);
(d)  cout<<((float (k/j) >=x) ? i+x:x-i);
(e)  cout<<(j+1 ==sqrt(j+k) ? x:k);
```

(2) على فرض أن

```
M=2;
N = M == 2  ? M++ : M+=50 ;
```

* ما هي قيمة المتغير N
* استخدم جملة if عوضا عن المؤثر الشرطي (?)

(3) باستخدام الجمل التالية:-

```
int a=2,b=4,c=8,d=5;
char x=1,y=0;
```

اوجد قيمة المتغير r لكل من الفقرات التالية :-

(a)

```
if (a)
c=a+b;
c+=d - b;
float r=c*d/float(b);
```

(b)

```
if(x || y && x)
    c/=b*a;
else
    c%=c*a;
int r=c+d;
```

(c)

```
if( (a-1)*a==a && a+b == c-d )
    b=a+c*c;
else
    b=c+a*b;
int r=b*a;
```

(4) اكتب برنامجا لادخال قيمتين مع طباعة القيمة الكبرى يتبعها الرسالة is larger وطباعة الرسالة

These numbers are equal

في حالة تساوي القيمتين باستخدام جملة if والمؤثر الشرطي.

(5) باستخدام جملة if مرة وجملة switch مرة أخرى، احسب قيمة a حيث:-

a=3x+10 اذا كان op=1

a=2x+3y+4 اذا كان op=2

a=25 اذا كان op=3

a=0 اذا كان غير ما ذكر اعلاه

(6) اكتب برنامجا مهمته مطالبة المستخدم بادخال ثلاث قيم مع ايجاد المجموع والمتوسط واصغر قيمة واكبر قيمة على ان تكون المخرجات في حالة ادخال القيم 12,25,23 مشابهة للأتي :-

Input three numbers please ==>: 12 25 23
sum is 60, Product is 6900, Average is 20
Smallest 12 while Largest 25

(7) باستخدام جملة switch أولا والمؤثر الشرط ثانيا، اقرأ قيمة المتغير x ثم أوجد قيمة المتغير y حيث:

$$y = \begin{cases} x+10 & \text{if } x>0 \\ 1000 & \text{if } x=0 \\ 25-x & \text{if } x<0 \end{cases}$$

(8) اكتب برنامجا لقراءة سطر يحتوي على قيمة حقيقية f وقيمة صحيحة n وقيمة حرفية ch ويحسب ويطبع العمليات f+n, f-n , f*n , f/n اعتمادا على قيمة الحرف ch الذي قد يأخذ المؤثرات +,-,*,/ أو تطبع الرسالة المناسبة اذا لم يكن الحرف احد المؤثرات الاربعة المذكورة وذلك باستخدام جملة if مره وجملة switch مره اخرى.

(9) اكتب برنامجا يقرأ مرتب البائع ومقدار مبيعاته ويحسب قيمة مكافأته المئوية التي هي على النحو التالي:-

2% اذا كانت مبيعاته اقل من أو تساوي ثلاثة اضعاف مرتبه.
3% اذا كانت مبيعاته اكبر من ثلاثة اضعاف مرتبه.
4% اذا زادت مبيعاته علي خمسة اضعاف مرتبه.

(10) باستخدام جملة switch، أكتب برنامجا يقرأ رقم الطالب ودرجته ويطبع الرقم والدرجة مع تقديره استنادا على الآتي:-

الحالة	الدرجة
Fail	اصغر من 50
Pass	اكبر من أو تساوي 50 وأصغر من 65
Good	اكبر من أو تساوي 65 وأصغر من 75
Very Good	اكبر من أو تساوي 75 وأصغر من 85
Excellent	اكبر من أو تساوي 85 وأصغر من 100

(11) تتبع واستنتج مخرجات البرامج التالية :-

a)

```
#include<iostream.h>
main()
{
      int x=3,y=6;
      if((x>4) && (y>x))
            y+=x-y;
      else
            if((y>x) && ( y/2 ==3))
                  y+=y+x;
            else
                  y+=x*y;-
      cout<<"Y="<<y<<endl;
}
```

b)

```
#include<iostream.h>
main()
{
      {
            int a=4,b=3;
            a+=b++;
            b=--a+2;
            float c=5;
            cout<<"A="<<a<<"  B="<<b<<"  C="<<c<<endl;
      }
      float a=4.5,b=3.6;
      if(b/2 == a/3)
```

```
	{
		int c=int(a/3+b);
		cout<<"A="<<a<<"  B="<<b<<"  C="<<c<<endl;
	}
	else
	{
		char a='A';
		int b=25;
		if ( b*3-10==a)
		{
			char c=a+2;
			b=b%11%6;
			cout<<"A="<<a<<"  B="<<b<<"  C="<<c<<endl;
		}
	}
	return 0;
}
```

(12) اكتب برنامجاً لاستقبال درجة الموظف ومرتبه الاساسي وضريبة الدخل حسب الآتي:-

الدرجة	المرتب الاساسي	ضريبة الدخل
7	200	%5
8	250	%10
9	300	%15
10	350	%20
13-11	400	%25
15-14	450	%30

وحساب صافي مرتبه حيث:

المرتب الصافي = المرتب الاجمالي - ضريبة الدخل

والمرتب الاجمالي = المرتب الاساسي + المستحقات

والمستحقات = مجموع العلاوات (علاوة العائلة = 10 دينارات، علاوة التمييز = 40 دينار)

(13) قم بكتابة برنامجا كاملا لإدخال ثلاثة امتحانات مع إيجاد متوسط اكبر امتحانين ثم طباعة كلمة pass اذا كان المتوسط اكبر من أو يساوي 50 وطباعة كلمة fail اذا كان غير ذلك .

(14) تتبع ما ينتج عنه البرنامج التالي في حالة إدخال القيمتين 3, 5 للمتغيرين b, a أولا والقيمتين 6, 2 ثانيا .

```
#include<iostream.h>
main()
{
        int a,b;
        cin>>a>>b;
        if(b % a ==b)
          if(( a>b) && (b != a))
            { a=2*b-3;  b +=a-1; }
         else
            a -=b;
         b=2*a+b;
         cout<<"A="<<a<<"  B="<<b;
        return 0;
}
```

الفصل الخامس: جمل التكرار والتفرعات

تحت شرط معين يتحتم علينا أحيانا تنفيذ جملة أو عدد من الجمل عدة مرات، ومن ناحية اخرى يتعين على المبرمج في العديد من المسائل تحويل مسار برنامجه الى جملة أو عدة جمل أو الخروج نهائيا من البرنامج، عليه سوف نتناول في هذا الفصل جمل التكرار Repetition Statements والجمل التفرعية Branching Statements بشيء من التفصيل.

1.5) جملة بينما The while Statement

لغة سي++ كغيرها في معظم اللغات الاخرى غنية جدا بالاوامر التي عن طريقها يتم تكرار تنفيذ جملة أو مجموعة من الجمل عدة مرات ومنها جملة while التي لها الشكل التالي :-

```
while(Logical_Expression)
Statement_1;
Statement_2;
```

هنا الجملة statement_1 تنفذ ويتكرر تنفيذها طالما أن الشرط Logical_Expression صحيحاً.

مثال 5-1) البرنامج الموالي مهمته تكرار وتنفيذ جملة الطباعة عدد 5 مرات .

```
include<iostream.h>
main()
{
      cout<<"The output of m as follwing"<<endl;
      int m=1;                    // initialization
      while(m <= 5)
      {
          cout<<" M="<<m;
          m+=1;                   // m as  increment
      }
      return 0;
}
```

في بداية البرنامج، تم تخصيص القيمة 1 الى للعداد m قبل الدخول الى حلقة التكرار، يأتي بعده الشرط m<=5 وحيث أنه صحيح لذلك تم تنفيذ الجمل الواقعة بين القوسين {} وهي زيادة قيمة العداد m بالقيمة 1 في كل مرة طالما كان الشرط صحيحا، وعندما تصبح قيمة m أكبر من 5 ينقل التحكم الى الجملة الموالية لقوسي الفئة أي نهاية البرنامج ، وعند تنفيذ هذا البرنامج فسوف يطبع الناتج المشابه للصورة التالية:-

```
The output of m as follwing
    M=1    M=2    M=3    M=4    M=5
```

اما اذا ما استبدل شرط بينما ليأخذ الشكل التالي:-

```
while(1)
```

فان البرنامج سوف يطبع القيم السابقة بنفس التسلسل ولكن بدون توقف لان حلقة التكرار لا تنتهي الا بوجود احد جمل التفرعات أو بالضغط على المفتاحين Ctrl-Z للخروج منها، ويستعمل هذا النوع من الشرط في حالة ما اذا كان المبرمج يريد تجربة برنامجه من حيث الادخال والمعالجة والاخراج.

مثال 5-2) ما هو ناتج تنفيذ البرنامج التالي؟

```
#include<iostream.h>
main()
{
       cout<<"The output of m as follwing"<<endl;
       int m=0;                    // initialization
       while(m <= 5)
       cout<<" M="<<m;
       m+=1;
       return 0;
}
```

الجواب: هو أن الناتج يكون غير الذي نتوقعه وهو الآتي:-

The output of m as following

M=1   M=1   M=1   ...

والسبب أن جملة while من المفروض ان يتم تنفيذها عددا من المرات حتي يصبح شرطها خاطئا وهذا لن يحدث ابدا، حيث تم تكرار جملة الطباعة فقط بينما لم تتم الزيادة المطلوبة لقيمة m لان جملة الزيادة خارجة عن نطاق حلقة التكرار لعدم ضم الجملتين بين قوسي الفئة {}، وعليه سوف لن يتجاوز هذا العداد القيمة 5 وينتج عنه طباعة M=1 الى مالا نهاية.

مثال 5-3) في البرنامج الموالي تم استخدام ما يعرف بالجملة الفارغة (Empty Statement) مع جملة while .

```
#include<iostream.h>
main()
{
       short i=0;
       while(++i <= 5)
       ;                           // empty statement
       cout<<"value of i = "<<i;
       return 0;
}
```

في هذا البرنامج بعدما خصصت القيمة 0 للمتغير i جاءت جملة while الذي يعمل هذا المتغير كعداد فيها، جاء الشرط i++<=5 الذي يضم:-

زيادة قيمة العداد i بالعدد 1 مقارنة هذا العداد مع العدد 5 وهذه ميزة من ميزات هذه اللغة، وهي استخدام العداد من ضمن الشرط الموجود في حلقة التكرار، حيث كان الشرط هنا صحيحا، بالتالي تم تنفيذ الجملة الفارغة (لا شيء ينفذ) وذلك بوضع الفاصلة المنقوطة (;) بعد الشرط مباشرة وتستمر زيادة قيمة المتغير بالقيمة 1 حتي يتجاوز العداد i القيمة 5، حينها تنتهي حلقة التكرار ويطبع البرنامج قيمة المتغير i النهائية كالآتي:-

```
value of i=6
```

في البرامج السابقة تم استخدام طريقة العدادات للخروج من الحلقات التكرارية، وقد حان الوقت لتقديم بعض الامثلة لتوضيح كيفية الخروج من اي حلقة كانت وذلك بادخال البيانات من لوحة المفاتيح.

مثال 4-5) المطلوب كتابة برنامج لقراءة قيمة متغير من النوع الحرفي مع طباعته بالنظام العشري وايقاف البرنامج اذا تم ادخال الحرف (?) .

```
#include<iostream.h>
#include <iomanip.h>
main()
{
      char ch;
      cout<<"Enter any character ==>: ";
      cin>>ch;
      while(ch != '?')
      {
           cout <<setw(20)<<"The character ["<<ch<<"] = "
                <<(int)ch<<" in decimal"<<endl;
           cout <<"Enter any character ==>: " ;
           cin>>ch;
```

```
    }
    cout <<setw(20)<<"The charcter you give is ["<<ch<<"]";
    return 0;
}
```

في هذا البرنامج تمت قراءة المتغير ch حتي يمكن الدخول الى حلقة التكرار ومن تم تنفيذ جمل الطباعة مع قراءة المتغير ch مرة اخرى لغرض تغيير قيمة الشرط والخروج من هذه الحلقة عندما يكون الحرف المدخل هو الرمز (?) وفي حالة عدم وجود جملة القراءة هذه فان هذه الحلقة لن تنتهي ، واذا تم تنفيذ البرنامج ستكون المدخلات والمخرجات كالآتي :-

```
Enter any character ==>: A
The character [A] =65 in decimal
Enter any character ==>: 7
The character [7] =55 in decimal
Enter any character ==>: ?
The charcter you give is [?]
```

مثال 5-5) اكتب برنامجا كاملا للحصول على متوسط n من الاعداد الحقيقية. وللحصول على هذا المطلوب ، يمكن اتباع الخطوات الاتية:-

1) إشهار متغير من النوع الصحيح مع تخصيص قيمة 0 له وليكن العداد counter ومهمته التحكم في حلقة التكرار والتي عن طريقها يتم قراءة الاعداد والحصول على حاصل جمعها.
2) اشهار متغير من النوع الحقيقي مع تخصيص قيمة 0 له لحفظ حاصل جمع الاعداد المدخلة وليكن المتغير sum.
3) قراءة عدد القيم المراد حساب متوسطها ويمثلها المتغير n .
4) تكرار الجمل التالية:-
   * قراءة العدد الموجود بالقائمة عن طريق المتغير number.
   * اضافة قيمة number الى المتغير sum.

* زيادة 1 الى العداد counter.

طالما أن قيمة العداد counter لا تتعدي عدد القيم n.

5) للحصول على المتوسط average اقسم المجموع بالمتغير sum على عدد الارقام n .

6) اطبع المتوسط.

وفيما يلي البرنامج المعد لتنفيذ هذه الخطوات:-

```
#include<iostream.h>
//          This program calculate the average
//           of N numbers using while loop
main()
{
      int n,counter=0;
      float number,average,sum=0;
      cout<<"Type the size of the list ==>: " ;
      cin>>n;
      while(counter++<n)
      {
            cout<<"Enter your number "<<counter<<" now ====>: ";
            cin>>number;
            sum+=number;
      }
      average=sum/n;
      cout<<"The average of all numbers = "<<average;
      return 0;
}
```

عند التنفيذ وادخال البيانات كالآتي:-

```
Type the size of the list ==>: 5
Enter your number 1 now ====>:10.0
Enter your number 2 now ====>:12.5
Enter your number 3 now ====>:-5
Enter your number 4 now ====>:9.9
Enter your number 5 now ====>:-1.0
```

سوف يكون المتوسط لهذه الاعداد هو السطر التالي:-

The average of all numbers = 5.28

قد نحتاج في بعض الاحيان الى استخدام اكثر من حلقة تكرار while وتكون متداخلة مع بعضها البعض في البرنامج الواحد بحيث كل واحدة منها تكون بداخل الاخري وتعرف (Nested while Statement)

الشكل العام

```
while(Logical_Expression_1)        // شرط الحلقة الخارجية
{
    Statement_1_1;       // جملة تابعة للحلقة الخارجية
    while(Logical_Expression_2)   // شرط الحلقة الداخلية
    {
        Statement_2_1;
        Statement_2_2;
            . . .
        Statement_2_Last;
    }
    Statement_1_2;        // جملة تابعة للحلقة الخارجية
}
```

مثال 5-6) المطلوب كتابة برنامج كامل مهمته حساب قيمة المتغير Y التي تساوي X مرفوعة للقوة N أي المعادلة التالية:-

$Y = X^N$

حيث المتغير X له القيمة الثابته 5 والمتغير N له القيم 6,5,4,3,2,1

```
#include<iostream.h>
main()
{
        int X=5, N=0;
        while(++N <=6)     // N  is a counter
        {
             long Y=1 ;
             int k=0;
             while(++k <=N)  // k  a counter
             Y*=X;
             cout<<X<<" to the power "<<N<<" =====> "<<Y << endl;
        }
        return 0;
}
```

في هذا البرنامج توجد حلقتا تكرار هما:-

1) الحلقة الاولي وهي تمثل الحلقة الخارجية ومهمتها تكرار الجملة المركبة التي تقع بين القوسين { } والتي تضم الاعلان عن بعض المتغيرات وجملة التكرار الثانية طالما ان المتغير N الذي يعمل كعداد لهذه الحلقة لم يتجاوز القيمة 6.

2) الحلقة الثانيه وهي تمثل الحلقة الداخلية ومهمتها حساب قيمة المعادلة

Y* = X;

أي ايجاد قيمة التعبير X مرفوعا للقوة N عن طريق المتغير k الذي يعمل كعداد لهذه الحلقة ويستمر تنفيذها طالما أن قيمة المتغير k لم تتجاو قيمة المتغير N، وفيما يلي نتائج هذا البرنامج بعد تنفيذه.

```
5 to the power 1 =====> 5
5 to the power 2 =====> 25
5 to the power 3 =====> 125
5 to the power 4 =====> 625
5 to the power 5 =====> 3125
5 to the power 6 =====> 15625
```

2.5) جملة افعل The do Statement

جملة do مشابة لجملة while ولكن تختلف عنها في أمرين أولهما ان جملة do تبدأ بتنفيذ الجمل التابعة لها ويتم التحقق من الشرط في أسفل هذه الجملة ، وثانيهما لابد من تنفيذ الجمل التي تلي جملة do ولو مرة واحدة على الاقل حتي ولو كان الشرط لم يتحقق ، وعليه يجب اخذ الحذر عند استخدام هذه الجملة.

الشكل العام

```
do
{
      Statement_1;
      Statement_2;
      . . .
      Statement_Last;
} while(Logical_Expression);
```

لاحظ أنه يجب وجود القوسين { } في حالة تنفيذ أكثر من جملة، وان ينتهي الشرط بالفاصلة المنقوطة (;).

مثال 7-5) البرنامج التالي يبين اول استخدامات جملة do والذي مهمته الحصول على مجموعة من القيم الصحيحة غير المعروف عددها وطباعتها.

```
#include<iostream.h>
main()
{
      int number,sum=0;
      char response;
```

```
    do
    {
        cout<<"Type the number please ==>: ";
        cin>>number;
        sum +=number;
        cout<<"any more number? ";
        cin>>response;
    }
    while( (response !='N') || (response !='n') );
    cout<<"Sum of numbers you gave = "<<sum;
    return 0;
}
```

في هذا البرنامج وبعد مجموعة من الاشهارات جاء الامر افعل do الذي يليه عدد من الجمل التنفيذية محصورة بين قوسي الفئة وتنتهي بالجملة while حيث يستمر تكرار تنفيذ هذه الجمل طالما ان قيمة التعبير المنطقي لشرط بينما وهو (response !='N') || (response !='n') ما يزال صحيحا ، ولايقاف تنفيذ هذا البرنامج، وجب ادخال الحرف n أو الحرف N ليصبح الشرط خاطئا.

عند تنفيذ هذا البرنامج ، سيكون هناك نوع من التحاور بين البرنامج ومنفذ هذا البرنامج، يبدأ برسالة تطالب بادخال قيمة صحيحة ولتكن 5 مثلا

Type the number please ==>: 5

وبعد اضافة هذه القيمة الى المجموع يظهر السؤال التالي

any more number?

أي الاستمرار في التحاور من عدمه، وهكذا حتي يتم الرد بإدخال الحرف المناسب وبالتالي طباعة مجموع القيم التي تم ادخالها.

مثال 8-5) المطلوب التعديل في البرنامج المكتوب بالمثال (5-5) بحيث ينتج عنه متوسط كل من الاعداد السالبة والاعداد الموجبة.

```
#include<iostream.h>
main()
{
	int n,counter=1; int negt=0,postv=0;
	float number,averg_n,averg_p;
	float sum_n=0,sum_p=0;
	cout<<"Type the size of the list ==>: " ;
	cin>> n;
	cout<<"Enter data ==>: ";
		do
	{
		// enter the value and add it to sum
		cin>>number;
		if (number<0)
		{
			negt+=1;
			sum_n+=number;
		}
		else
		{
			postv+=1;
			sum_p+=number;
		}
	} while(counter++<n);
	averg_n=sum_n/negt;
	averg_p=sum_p/postv;
	cout<<"The average of positive numbers = "<<averg_p<<endl;
	cout<<"The average of negative numbers = "<<averg_n;
	return 0;
}
```

هنا تتكرر الجملة المركبة التي بين قوسي الفئة طالما ان العداد counter الذي يبدأ بالقيمة 1 هو اصغر من عدد القيم المراد ادخالها n ، وعند تنفيذ هذا البرنامج ، يجب الاخذ في الاعتبار ادخال قيمة موجبة وقيمة سالبه على الاقل حتى يتم تفادي عملية القسمة التي تلي جملة while والا سينتج عن هذا البرنامج الرسالة divide error أي ان هناك القسمة على الصفر ، واذا ما نفذ هذا البرنامج بادخال البيانات المناسبة التالية :-

Type the size of the list ==>: 7

Enter data ==>:10 -5 12 -9 2 -7 8

The average of positive numbers = 8

The average of negative numbers = -7

مثال 5-9) تداخل جملة do يمكن استخدامها وتوضيحها في البرنامج الموالي.

```
#include<iostream.h>
main()
{
	int inner,outer=1;
	do
	{
		cout<<endl<<"OUTER DO STATEMENT "<<endl;
		inner=1;
		do
		{
			cout <<"inner do statement = "<< end;
		} while(++inner<5);
	} while(++outer<=2);
	return 0;
}
```

في هذا البرنامج استخدمت جملتا do، الاولي ومهمتها طباعة الرسالة

OUTER DO STATEMENT

مع تنفيذ جملة do الثانية 5 مرات والتي مهمتها طباعة الرسالة

inner do statement

3.5) جملة لاجل The for Statement

جملة for تعتبر احدي اهم جمل التحكم والتكرار في هذه اللغة، وهي تستخدم عندما يكون المبرمج على علم مسبق بتكرار جملة أو مجموعة من الجمل المركبة لعدد محدود من المرات.

تأخذ جملة for الشكل العام التالي :-

```
for(expression1;expression2;expression3)
      statement;
next statement;
```

حيث:

expression1 القيمة الابتدائية التي تحدد للمتغير على أنه عداد.

expression2 شرط استمرار حلقة التكرار.

expression3 جملة الزيادة أو النقصان للعداد في دليل الحلقة.

ويتم تنفيذ التعبير الاول expression1 الذي مهمته تخصيص القيمة الابتدائية لعداد حلقة التكرار، يلي ذلك تنفيذ التعبير الثاني expression2 الذي يمثل شرط الحلقة، فاذا كان الشرط لا يساوي صفرا (صحيح)، عندها تنفذ الجملة statement ثم تتم زيادة أو نقصان التعبير الثالث expression3 ويعاد بعدها اختبار الشرط مجددا وهكذا حتى يصير التعبير الثاني expression2 خطأ عندها ينتقل التحكم الى الجملة التالية وهي next statement.

أما في حالة إحتواء جملة for لأكثر من جملة، عندها يجب ضمها بين قوسي الفئة.

مثال 5-10) يمكن استخدام جملة for لتكرار وتنفيذ جملة واحدة كما يوضحه البرنامج التالي:-

```
#include<iostream.h>
main()
{
      for(int m=1;m<=7;m++)
            cout<<"  M = "<<m;
      return 0;
}
```

الشرح: التعبير الاول int m=1 أي الاعلان عن المتغير m كعداد للحلقة وتخصيص القيمة الابتدائية 1 له.

التعبير الثاني m<=7 يمثل شرط الحلقة، ويعني نفذ الجملة الموالية لجملة for أي طباعة قيمة المتغير m طالما أن قيمة العداد m لم تتجاوز العدد 7.

التعبير الثالث m++ يعني زيادة قيمة العداد m بالقيمة 1 بعد كل خطوة تنفذ فيها جملة الطباعة.

وأخيرا فسيتم طباعة الآتي عند تنفيذ هذا البرنامج.

M=1 M=2 M=3 M=4 M=5 M=6 M=7

واذا ما غيرت جملة for في البرنامج السابق بالصورة التالية :-

```
for(int m=7;m>=1;m--)
```

سوف ينتج عند تنفيذه طباعة النتائج السابقة تنازليا وهي تشبه الآتي :-

M=7 M=6 M=5 M=4 M=3 M=2 M=1

وهذا يعني :-

خصص العدد 7 للمتغير m طالما أن المتغير m مايزال اكبر من أو يساوي 1 كرر جملة الطباعة مع طرح العدد 1 من المتغير m.

خذ مثلا جملة for التالية:-

```
for(int m=1;++m<=7;)
```

حيث استخدم فيها التعبير

```
++m<=7
```

وهو يعني اضافة القيمة 1 للعداد m مع مقارنته بالقيمة 7 في كل مرة تنفذ فيها هذه الجملة، لاحظ استخدام الفاصلة المنقوطة بعد هذا التعبير وهذا يعنـي ان التعبير الثالث قد تم الاستغناء عنه.

مثال 5-11) اكتب برنامجا لايجاد مجموع الاعداد

sum=4.5+5.0+5.5+...+10.0

يمكن حل هذه المسألة بطريقتين :-

الطريقة الاولي :-

```
#include<iostream.h>
 main()
{
        float sum=0.0;
        for(float a=4.5;a<=10;a+=.5)
              sum+=a;
        cout<<"THE SUM IS  "<<sum;
        return 0;
}
```

في هذه الطريقة استخدم:-

- التعبير الاول a=4.5 ويمثل البداية بالقيمـة العدديـة الحقيقيـة float وهـذا مسموح به في هذه الحلقة.
- التعبير الثاني a<=10 ويمثل شرط هذه الجملة.
- التعبير الثالث a+=0.5 وهي زيادة المتغير a بالقيمة 0.5

الطريقة الثانية :-

```
#include<iostream.h>
main()
{
        float sum=0.0;
        for (float a=4.5;a<=10;)
        {
             sum+=a;
             a+=0.5;
        }
        cout<<"THE SUM IS  "<<sum;
        return 0;
}
```

هنا تم استخدام مقدار الزيادة وهو التعبير a+=0.5 من ضمن الجملة المركبة لجملة for بعد تخزين القيمة الابتدائية 4.5 في المتغير sum نلاحظ في الطريقتين أنه يمكن استخدام التعبيرات الثلاثة المستخدمة مع جملة for كمتغيرات من النوع الحقيقي، والناتج في كليهما هو الآتي:-

THE SUM IS 87

مثال 5-12) يمكن الحصول على الناتج في المثال الاخير باستعمال مؤثر الفاصلة والجملة الفارغة.

```
#include<iostream.h>
main()
{
        for (float sum=0.0,a=4.5;a<=10;sum +=a,a+=0.5)
        ;                         // empty ststement
        cout<<"THE SUM IS "<< sum;
        return 0;
 }
```

نلاحظ هنا :-

- استخدام مؤثر الفاصلة في بداية الحلقة for ومهمته الاعلان عن المتغيرين a , sum واسناد القيمتين 0 , 4.5 لهما على التوالي.
- استخدام مؤثر الفاصلة في آخر الحلقة ومهمته إضافة قيمة المتغير a إلى المتغير sum في كل مرة يزداد فيها المتغير a بالقيمة العددية 0.5 .
- تلا جملة for استخدام الجملة الفارغة بوضع الفاصلة المنقوطة (;) وعليه سوف يعطي هذا البرنامج نفس النتائج السابقة تماما.

مثال 5-13) بالامكان استخدام التعبيرات الثلاثة في جملة for كمتغيرات حرفية، البرنامج التالي يبين هذا الاستخدام، حيث يقوم هذا البرنامج بطباعة الحروف الهجائية تنازليا أي من الحرف Z الى الحرف A

```
#include<iostream.h>
 main()
{
      for(char ch='Z';ch>='A';--ch)
            cout<< ch;
      return 0;
}
```

مثال 5-14) اكتب برنامجا كاملا مهمته حساب حاصل جمع الاعداد الزوجية الصحيحة المحصورة بين القيمة 2 والقيمة 10.

```
#include<iostream.h>
main()
{
      int sum=0;
      cout<<endl<<" THE SUM OF EVEN NUMBERS ==> " ;
      for(int i=2;i<=10;i+=2)
      {
            sum +=i;
            cout<<"  "<<i;
```

```
    }
    cout<<" = "<<sum;
    return 0;
}
```

بالبرنامج جملة for التي تحتوي على :-

1) التعبير الاول i=2 حتي تكون البداية من العدد 2 حسب المطلوب.

2) ولكي نتحصل على الاعداد الزوجية يجب ان يكون مقدار زيادة المتغير i بمقدار 2، اما بالشكل i+=2 أو الشكل i=i+2 ثم تنفيذ الجملة المركبة 5 مرات بقصد طباعة قيمة المتغير i وهي 2 4 6 8 10 مع حاصل جمع هذه القيم.

في جملة for يجوز الاستغناء عن تعبير البداية أو النهاية أو الزيادة والنقصان أوحذفها جميعا بوضع فواصل منقوطة بدلا من هذه التعابير.

مثال 5-15) ماذا يحدث اذا ما تم الاستغناء عن التعبير الاول في جملة for؟، لتوضيح هذا السؤال يمكن اعادة كتابة البرنامج في المثال (5-14).

```
#include<iostream.h>
main()
{
    int sum=0;
    cout<<endl<<" THE SUM OF EVEN NUMBERS ==> " ;
    int i=2;
    for(;i<=10;i+=2)
    {
        sum +=i;
        cout<<"  "<<i;
    }
    cout<<" = "<<sum;
    return 0;
}
```

في هذا البرنامج تم تعريف وتخصيص القيمة 2 للمتغير i قبل تنفيذ جملةfor وبدلا عن هذا التعبير وضعت الفاصلة المنقوطة داخل الحلقة ، وسوف يعطي البرنامج نفس النتائج في المثال المشار إليه.

مثال 5-16) البرنامج الموالي يوضح استخدام جملة for لانهائية.

```
#include<iostream.h>
main()
{
        int i=2;
        for( ; ; )
        {
             cout<<"  "<<i;
             i+=2;
        }
        return 0;
}
```

في غياب الشرط سوف ينتج عنه طباعة قيمة المتغير i بداية من 2 إلى مالانهاية.

مثال 5-17) المطلوب كتابة برنامج كامل لحساب قيمة المتغير Z

Z=(X-Y)/(X+Y)

حيث المتغير X يأخذ القيم X=1.0,1.5,2.0,...,5.0

والمتغير Y يأخذ القيم Y=-5.0,-4.5,-4.0,...,-1.0

بشرط استخدام جملة for للحصول على قيمتي المتغيرين Xو Y .

الحل: للحصول على قيمة المتغير Z ينبغي اتباع الآتي :-

1) تخصيص القيمة الابتدائية 1.0 للمتغير الحقيقي X.

2) تخصيص القيمة الابتدائية 5.0- للمتغير الحقيقي Y.

3) اذا كانت قيمة المقام X+Y لاتساوي 0 عندها احسب قيمة المتغير Z واطبع كلا من المتغيرات Z,Y,X أما اذا كانت غير ذلك تغاض عن حساب قيمة Z لان قيمة التعبير هنا غير معرفة مع ذكر السبب، وفي كلا الحالتين اضف المقدار 0.5 لكل من المتغيرين Y,X .

4) كرر هذه الخطوات حتى تكون قيمة X أكبر من 5 وقيمة Y أكبر من 1.0-.

والبرنامج التالي يبين تنفيذ هذه الخطوات.

```
#include <iostream.h>
#include <iomanip.h>
main()
{
      cout<<" X      Y    Z"<<endl<<endl;
      for(float x=1.0, y=-5.0;x<=5.0 && y<=-1.0;x+=0.5,y+=0.5)
      {
            if( x+y ==0)
            {
                  cout <<"Z undefine at X="<<x
                        <<" and Y="<<y<<endl;
            }
            else
            {
                  float z=(x-y)/(x+y);
                  cout <<setiosflags(ios::showpoint | ios::fixed)
                        <<setprecision(2)<<x<<"   "<<y<<"   "<<z<<endl;
            }
      }
      return 0;
}
```

هنا تم استخدام جملة for بالصورة الجديدة التالية:-

```
for(float x=1.0, y=-5.0;x<=5.0 && y<=-1.0;x+=0.5,y+=0.5)
```

وهي تضم:-

1) مؤثر الفاصله في بداية الحلقة والذي مهمته الاعلان عن المتغيرين y,x وتخصيص القيمتين 5,1- لهما على التوالي، وبالتالي اعتبرا كعدادين للحلقة نفسها.

2) تكرار الجمل التابعة لحلقة التكرار اذا تحقق شرطها وهو

x<=5.0 && y<=-1.0

3) يأتي مؤثر الفاصلة في مؤخر الحلقة وهو زيادة قيمة المتغيرين y,x بالقيمة 0.5

اخيرا اذا ما تم تنفيذ البرنامج السابق سيكون الناتج مشابها للآتي:-

```
X     Y      Z
1.00  -5.00  -1.50
1.50  -4.50  -2.00
2.00  -4.00  -3.00
2.50  -3.50  -6.00
Z undfine at X=3.00 and Y=-3.00
3.50  -2.50  6.00
4.00  -2.00  3.00
4.50  -1.50  2.00
5.00  -1.00  1.50
```

لم تطبع قيمة المتغير Z عندما يكون المتغير X يساوي 3.0 والمتغير Y يساوي 3.0- لان X+Y تساوي 0 بل تمت طباعة الرسالة التي تدل على أن قيمة Z غير معرفة.

توجد ايضا جملة for المتداخلة (Nested for Statement) والتي يكثر استخدامها خصوصا في معالجة المصفوفات وغيرها، وشكل هذه الجملة هو الآتي:-

```
for(...)
    for(...)
        statement;
```

بحيث تكون كل جملة داخل الاخرى.

مثال 5-18) في البرنامج التالي تم استخدام أكثر من جملة for المتداخلة مع المتغيرات من النوع الحرفي، والمطلوب استنتاج ما الذي يطبعه هذا البرنامج عند تنفيذه ؟.

```
#include<iostream.h>
main()
{
	for(char l='A';l<='C';l++)
		for(char m='A';m<='C';m++)
		   if(l!=m)
		      for(char n='A';n<='C';n++)
			  if((l!=n)&&(m!=n))
			      cout<<l<<m<<n<<"\n" ;
	return 0;
}
```

مثال 5-19) فصل به عدد من الطلبة، المطلوب كتابة برنامج كامل مهمته ادخال رقم قيد الطالب مع اربعة امتحانات في مادة الحاسب الآلي مع طباعة رقم القيد ومتوسط هذه الامتحانات لكل طالب.

```
#include<iostream.h>
// Calculate the average of student grades using nested for
main()
{
	int number_of_student;
	long student_number;
	short mark; float average;
	cout<<"Enter number of student ==>: ";
	cin>>number_of_student;
	cout<<"------------------------";
```

```
        // Outer for statement
    for(short i=1;i<=number_of_student;i++)
    {
        cout<<"\n Type student#"<<i;
        cout<<" and 4 grades ";
        cin>>student_number;
                // Inner for statement
        float sum=0.0;
        for(short j=1;j<=4;j++)
        {
                cin>>mark;
                sum+=mark;
        }
        average=sum/4;
        cout   <<"\nThe student number "<<student_number
                <<" has the average "<<average;
    }
    return 0;
}
```

في هذا البرنامج تم استخدام:-

1) الحلقة for الاولى ومهمتها قراءة رقم قيد الطالب مع حساب متوسط درجات الامتحانات وطباعة النتائج المطلوبة لفصل به عدد 4 طلبة.

2) الحلقة for الثانية وظيفتها قراءة 4 درجات لكل طالب مع حساب مجموع هذه الدرجات.

عند التنفيذ يرد الحاسب بالسؤال التالي:-

Enter number of student ==>:

وتعني ادخال عدد الطلبة المطلوب حساب متوسطهم، وعليه تم الرد على هذا السؤال بادخال العدد 4 بالصورة التالية:-

Enter number of student ==>: 4

حينها يطالب الجهاز بادخال رقم قيد ودرجات الطالب الاول

Type student #1 and 4 grades: 9805001 60 70 50 80

عندها يرد الجهاز بالمعلومات المطلوبة منه كالآتي:-

The student number 9805001 has the average 65.00

وهكذا تستمر عملية ادخال البيانات واخراج المعلومات لبقية طلبة الفصل وقد تكون كما يلي:-

Type student #2 and 4 grades: 9805026 30 45 39 42

The student number 9805026 has the average 39.00

Type student #3 and 4 grades: 9805093 55 55 55 55

The student number 9805093 has the average 55.00

Type student #4 and 4 grades: 9805999 75 69 81 89

The student number 9805999 has the average 78.50

4.5) جملة اذهب الى The goto Statement

تستخدم هذه الجملة لتغيير مسار البرنامج التتابعي أو الخروج من الحلقات التكرارية مثل while و for وغيرها واحيانا الخروج نهائيا من البرنامج، وشكلها العام هو:-

```
goto lable;
 . . . . . .
 lable:statement;
```

حيث lable اسم العنوان الذي يمكن وضعه في أي مكان من البرنامج الرئيسي أو الفرعي، ويكون ذا اسم فريد، وهذه الجملة قد تنفذ تحت شرط معين وتسمي if المشروطة، أو بدون شرط وتعرف بجملة if غير المشروطة.

ينبغي التحفظ والتقليل من استعمال هذه الجملة في البرامج لانها في بعض الاحيان تؤدي الي جعل البرنامج غير مفهوم وغير واضح هيكليا وخصوصا وقت المراجعه والتصحيح.

مثال 5-20) اكتب برنامجا لقراءة عدد لا يزيد عن 10 قيم صحيحة موجبة، ثم أوجد مجموع هذه القيم، مع ايقاف تنفيذ البرنامج عند ادخال قيمة سالبة أو صفرية.

```
#include<iostream.h>
main()
{
        int number,posnumber=0;
        cout<<"Please type 10 values :"<<endl;
        for(int i=1;i<=10;i++)
        {
              cout<<"Enter value" <<i<<" ==>: ";
              cin>>number;
              if (number<=0)
              {
                      cout<<"This is negative or zero number"<<endl;
                      goto last;
              }
              posnumber+=number;
        }
        last: cout<<"The sum of positive values are : "<<posnumber;
        return 0;
}
```

في هذا البرنامج لم يتم ادخال القيم العشرة عن طريق الجملة cin والموجودة في جملة for، والسبب هو شرط جملة if فبمجرد ادخال قيمة غير موجبة يتحقق الشرط (number <=0) واصبحت قيمته صحيحة، عندها تطبع الرسالة التي تحقق الشرط مع انتقال التحكم الى جملة الموجودة أمام العنوان last يتبعها النقطتان (:) حيث ينتج عنها طباعة مجموع القيم الموجبة فقط.

عند تنفيذ هذا البرنامج ستظهر الرسالة الدالة على ادخال 10 قيم الواحدة تلو الاخري كما يلي :-

```
Please type 10 values :
Enter value 1 ==>: 5
Enter value 2 ==>: 8
Enter value 3 ==>: 7
Enter value 4 ==>: 1
Enter value 5 ==>: -3
```

هنا القيمة الخامسة كانت قيمة سالبه، عليه سوف يطبع البرنامج الرسالة الدالة على أن القيمة سالبه أو صفرية يليها مجموع القيم الموجبة فقط أي

```
This is negative or zero number
The sum of positive values are: 21
```

5.5) جملة أقطع The break Statement

تستخدم هذه الجملة للخروج من الحلقات التكرارية المختلفة حيث يتم انهاء التكرار متى وصل التنفيذ الى هذه الجملة.

مثال 5-21) يمكن اعادة كتابة البرنامج في المثال (5-20) باستخدام جملة break كالتالي:-

```
#include<iostream.h>
main()
{
	int number,posnumber=0;
	cout<<"Please type 10 values :"<<endl;
	for(int i=1;i<=10;i++)
	{
		cout<<"Enter value" <<i<<" ==>: ";
		cin>>number;
		if(number<=0)
		{
```

```
            cout<<"This is negative or zero number"<<endl;
            break;
        }
        posnumber+=number;
    }
    cout<<"The sum of positive values are : "<<posnumber;
    return 0;
}
```

في هذا البرنامج عند ادخال قيمة عددية سالبة أو صفرية، يأتي تنفيذ جملة break التي اجبرت البرنامج على الخروج مـن الحلقـة التكراريـة for قبـل أن تنتهي بموجب جملة if أي عند تحقيق الشرط (numbe <=0) ويتـم تنفيـذ جملـة الطباعة الموالية للشرط وينتقل سير التحكم الى الجملة الموالية لنهاية الحلقـه forوهي الجملة التي تطبع مجموع القيم الموجبة التي تم ادخالها.

6.5) دالة الخروج The exit() Function

وهي تعني الخروج من البرنـامج كليـا كمـا يـدل اسـمها ، وترجـع بالقيمـة صفرا اذا كان البرنامج نفذ على احسن ما يرام ، وترجع بالقيمة غير الصفـر اذا كـان هنـاك بعـض الاخطـاء ، ويسـتخدم ملـف العنـاوين <stdlib.h> حتـي يتمكن المترجم من التعرف على هذه الدالة.

مثال 5-22) ماذا يحـدث اذا تمـت اعـادة كتابـة البرنـامج فـي المثـال (5-20) باستخدام دالة الخروج exit

```
#include<iostream.h>
#include <stdlib.h>  // for exit
main()
{
    int number,posnumber=0;
    cout<<"Please type 10 values :"<<endl;
```

```
    for(int i=1;i<=10;i++)
    {
        cout<<"Enter value" <<i<<" ==>: ";
        cin>>number;
        if(number<=0)
        {
            cout<<"This is negative or zero number"<<endl;
            exit(0);
        }
        posnumber+=number;
    }
    cout<<"The sum of positive values are : "<<posnumber;
    return 0;
}
```

عندما يكون شرط جملة if صحيحا تطبع الرسالة التي تدل على ان القيم المدخلة سالبة أو صفرية ثم يصل التنفيذ الى دالة الخروج ()exit التي تعني الخروج ليس من حلقة التكرار for بل تعني الخروج نهائيا من البرنامج وبالتالي لن تنفذ الجملة الموالية لجملة for وهي :

```
cout<<"The sum of positive values are : "<<posnumber;
```

ولن يطبع البرنامج مجموع الاعداد الموجبة المدخلة.

7.5) جملة الاستمرار The continue Statement

تختلف مهمة هذه الجملة عن الجمل التفرعية السابقة فهي تعني الاستمرار في توجيه التحكم الى نهاية الحلقة مثل for و while ، وبالتالي الرجوع الى بداية الحلقة واكمال تنفيذها حتي نهايتها.

مثال 5-23) البرنامج التالي وهو نفس البرنامج المكتوب بالمثال(5-20) يوضح كيفية استخدام جملة continue.

```
#include<iostream.h>
main()
{
	int number,posnumber=0;
	cout<<"Please type 10 values : "<<endl;
	for(int i=1;i<=10;i++)
	{
		cout<<"Enter value" <<i<<" ==>: ";
		cin>>number;
		if(number<=0)
		{
			cout<<"This is negative or zero number"<<endl;
			continue;
		}
		posnumber+=number;
	}
	cout<<"The sum of positive values are : "<<posnumber;
	return 0;
}
```

وقت التنفيذ وادخال البيانات المناسبة كالآتي:-

```
Please type 10 values :
Enter value 1 ==>: 3
Enter value 2 ==>: 10
Enter value 3 ==>: 5
Enter value 4 ==>: -6
This is negative or zero number
Enter value 5 ==>: 2
Enter value 6 ==>: 0
This is negative or zero number
Enter value 7 ==>: 20
Enter value 8 ==>: 5
Enter value 9 ==>: 2
Enter value 10 ==>: 8
The sum of positive values are: 55
```

وعند تحقيق شرط جملة if تطبع الرسالة الدالة على أن القيمة المدخلة قيمة سالبة أو صفرية ثم تنفذ جملة continue ويتحول بسببها سير التحكم الى القوس المغلق { والدال على نهاية الجملة for وبالتالي لن يتوقف ادخال بقية القيم بل يستمر حتي نهاية الحلقة ، ويتم بذلك الحصول على مجموع كل القيم الموجبة فقط.

8.5) تمارين Exercises

(1) باستخدام جملة for المطلوب طباعة الحروف من 'A' الى 'Z' حرفان في كل سطر مرة وخمسة حروف مرة أخرى.

(2) أكتب برنامجا لقراءة القيم التالية:

12	40
35	24
-7	50
30	30
20	-9

أوقف تنفيذ البرنامج اما بعدد السطور أو بادخال قيمة صفرية في نهاية هذه القيم مع حساب متوسط العمودين ومتوسط كل صف.

(3) كم مرة تطبع الرسالة "good luck" عن طريق الجزء التالي من البرنامج

```
for(int a=1;a<6;a+=2)
   for(int b=6;b>1;b-=2)
     for(int c=b;c<=3;c++)
       cout<<"good luck"<<endl;
```

(4) كم سطرا يطبع عن طريق الجملة مع التوضيح

```
for(float a=0; a!=10.0;a+=0.1)
     cout<<"A="<<a<<endl;
```

(5) باستخدام جملة while ، أكتب برنامجا لقراءة n ثم أوجد واطبع الآتي :-

(a) sum=1+2+3+...+(n+2)

(b) sum=1-2+3-4+...n

(c) sum=1-1/2+1/3-1/4+...1/n

(d) $sum=1+x+x^2/2!+x^3/3!+...+x^n/n!$

(6) أكتب برنامجا لحساب حاصل ضرب وجمع مربعات الاعداد الصحيحة الفردية الواقعة بين عددين صحيحين يتم ادخالهما عن طريق لوحة المفاتيح.

(7) باستخدام جملة do اكتب برنامج لقراءة المتغيرين m,n ، ثم يحسب قيمة المعادلة:

$$p = \frac{n!}{(n-m)!}$$

(8) أعد الجمل التالية باستخدام جملة while

(a)

```
for(int i=1;i<10;++i)
    cout<<"I ==>"<<i;
cout<<"I*I ==>"<<i*i<<endl;
```

(b)

```
    short i=1;
    do
    {
        cout<<"I ==>"<<i;
        if(i !=3)
        {
                i++;
                continue;
        }
        cout<<endl<<"Hi User"<<endl;
        i++;
    } while(i!=10);
    cout<<endl<<"all done";
}
```

(9) المطلوب كتابة برنامج مهمته قراءة ثلاث قيم من النوع الحقيقي تمثل طول اضلاع مثلث مع:

* طباعة Triangle في حالة مجموع أي ضلعين اكبر من قيمة الضلع الثالث.

* طباعة Equilateral في حالة تساوي الاضلاع.
* طباعة Isosceles في حالة تساوي ضلعين.
* طباعة Scalene في حالة اختلاف الاضلاع.

بحيث اذا تم ادخال القيم بالشكل التالي :-

Enter 3 lengths ==> 3 4 4

يكون الاخراج بالصورة المشابهة للآتي :-

Triangle is ==>True

Equilateral is ==> False

Isosceles is ==> True

Scalene is ==> False

(10) أكتب برنامجا مهمته ايجاد طول أي عدد صحيح موجب أو سالب ويعكسه، فمثلا العدد 12345- طوله 5 ويصبح 54321-

(11) اكتب برنامجا لطباعة جدول ضرب الاعداد من 1 الى 9 بحيث يكون بالشكل التالي:-

	1	2	3	4	5	6	7	8	9
1	1								
2	2	4							
3	3	6	9						
4	4	8	12	16					
5	5	10	15	20	25				
6	6	12	18	24	30	36			
7	7	14	21	28	35	42	48		
8	8	16	24	32	40	48	56	64	
9	9	18	27	36	45	54	63	72	81

(12) اذا كان لدينا البرنامج التالي:--

```
#include<iostream.h>
main()
{
	int a,b,c;
	for (int i=1;i<=4;i++)
	{
		cin>>a>>b;
		if(a*b<=b+b)
			goto done;
		{
			c=b-a;
			goto not_yet;
		}
		done: c=a+b;
		not_yet: cout<<c<<endl;
	}
}
```

المطلوب تتبعه واستنتاج المخرجات، بافتراض أن المتغيرات b,a كان لها القيم التالية :--

(a) 2 4 (b) 5 2
(c) 5 5 (d) -3 8

(13) ما هو ناتج البرامج التالية:--

(a)

```
#include <iostream.h>
main()
{
	int i=8;
	while(i>=1)
	{
		for(int j=i;j<=8;j++)
			cout<< (i % 2 ? "<==" : "==>");
		cout<<endl;
		i--;
	}
}
```

(b)

```
#include <iostream.h>
main()
{
        int i=0,j=0,k;
        do
        {
              j+=i*j;
              cout<<"J="<<j<<endl;
              do
              {
                     k=i+j*3;
                     cout<<"J="<<j<<"  K="<<k<<endl;
                     j+=1;
              } while (k<=4);
        } while (++i<3);
}
```

(14) فصل دراسي به عدد num من الطلبة أكتب برنامجا لقراءة رقم الطالب وجنسه ودرجته في ثلاث أمتحانات، أوقف تنفيذ البرنامج عند ادخال رقم الطالب سالبا، والمطلوب:-

* طباعة رقم الطالب ومتوسط امتحاناته والحرف المقابل لمتوسطه على النحو التالي:-

المتوسط	الحرف
اكبر من أو يساوي 85	A
اصغر من 85 واكبر من أو يساوي 75	B
اصغر من 75 واكبر من أو يساوي 65	C
اصغر من 65 واكبر من أو يساوي 50	D
اصغر من 50	F

* حساب عدد الطالبات الناجحات بالفصل.
* عدد التقديرات F,A بالفصل.

* المتوسط العام للطلبة.

* طباعة اكبر متوسط مع رقم القيد في هذا الفصل.

(15) تتبع البرنامج التالي واوجد ما الذي يطبعه، وما الذي يمكن استنتاجه من تغيير الجملة break بجملة continue مرة وبدالة exit مرة أخرى ؟.

```
#include <iostream.h>
#include <process.h>
main()
{
	int i=5,j;
	for(;;)
	{
		i-=2;
		cout<<"I="<<i<<endl;
		if(i<0)
			break;
		j=-2;
		for(;;)
		{
			j+=2;
			cout<<"J="<<j<<endl;
			if(j>i)
				break;
		}
		cout<<"inner"<<endl;
	}
	cout<<"outer"<<endl;
}
```

الفصل السادس: دوال معالجة الحرفيات

هذا الفصل يتناول كيفية التعامل مع البيانات سواء منها ذات الحرف الواحد أو السلسلة الحرفية ، حيث نقدم مجموعة من الدوال المختلفة التي تعالج مثل هذا النوع من البيانات.

1.6) دوال معالجة الحرف Character Manipulation Functions

البيانات التي هي من النوع الحرفي (Character) ويتعامل معها جهاز الحاسب يوجد لها في لغة سي++ مجموعة من الدوال مهمتها معالجة متغيرات من هذا النوع على أن يستخدم ملف العناوين <ctype.h> مع هذه الدوال التي يتم شرحها فيما بعد.

2.6) دوال اختبار الحرف Character Testing Functions

وتقوم هذه الدوال باختبار حرف واحد ، حيث ترجع بقيمة غير صفرية (true) اذا ما كان الحرف رقما أو حرفا أو فراغا أو علامة ترقيم وهكذا ، اما اذا كان غير ذلك فهي ترجع بالقيمة صفر (false) ، والجدول التالي يعرض البعض منها.

الدالة	معناها
isdigit(x)	هل x رقم ؟ (0-9)
isalpha(x)	هل x حرف أبجدي ؟
isalnum(x)	هل x حرف أو رقم ؟
isxdigit(x)	هل x في النظام الستة عشرى (0-F) ؟
islower(x)	هل x حرف صغير ؟
isupper(x)	هل x حرف كبير ؟
isspace(x)	هل x فراغ أو ('\n') أو ('\f') أو ('\r') أو ('\t') ؟
iscntrl(x)	هل x رمز تحكم ؟
ispunct(x)	هل x علامة ترقيم (أي عدا الارقام والحروف والفراغ) مثل الشارحة والفاصلة ؟
isprint(x)	هل x قابلة للطباعة بما فيها الفراغ ؟
isgraph(x)	هل x أحد الحروف المطبوعة ماعدا الفراغ ؟

مثال 6-1) وحتي يمكن شرح وفهم مهام بعض هذه الدول، البرنامج الموالي يستقبل متغيرين من النوع الحرفي، ثم ينتج عنه اختبارهما وبالتالي اظهار الرسالة المناسبة لهذين الحرفين.

```
        // Program using isdigit function
#include <iostream.h>
#include <conio.h>      // for clear screen
#include <ctype.h>     //  for isdigit function
main()
{
      char ch1,ch2;
      clrscr();
      cout<<" Enter two characters please ==>: ";
      cin>>ch1>>ch2;
      cout<<endl<<" According to isdigit function "<<endl;
      int answer=isdigit(ch1);
      cout <<ch1<< ( answer ? " is a " : " is not a ")
           <<"digit"<<endl;
```

```
    answer=isdigit(ch2);
    cout <<ch2<< ( answer ? " is a " : " is not a ")
         <<"digit"<<endl;
    getch();
    return 0;
}
```

في بداية هذا البرنامج توجد الدالة clrscr() والتي مهمتها مسح البيانات أو المعلومات الموجودة بشاشة العرض بقصد امكانية التخاطب والحوار على شاشة نظيفة بين البرنامج والمستخدم لهذا البرنامج بشرط استخدام ملف العناوين <conio.h> وقت استخدام هذه الدالة، وعليه عند التنفيذ تظهر الرسالة

Enter two characters please ==>:

على شاشة نظيفة لاعطاء المنفذ الفرصة للرد على هذه الرسالة بادخال حرفين 5& كالآتي

Enter two characters please ==>:5&

عندها سوف ينتقل التحكم الى اختبار هل الحرف الاول من النوع الرقمي أم لا، فاذا تحقق شرط المؤثر ? ، عندها يرد الحاسب بطباعة الرسالة الدالة على ان هذا الحرف هو من النوع الرقمي كالآتي:-

5 is a digit

والا فيطبع غير ذلك

& is not a digit

نلاحظ ان الحرف الاول 5 هو من ضمن الارقام، في حين أن الحرف الثاني & هو غير ذلك، ووظيفة الدالة isdigit() مكافئة للتعبير المنطقي

((ch1>='0') && (ch1<='9'))

3.6) دوال تبديل الحرف Character Conversion Functions

هناك عدد من الدوال مهمتها القيام بتبديل الحرف من صغير الى كبير وبالعكس، وهذه الدوال موجودة بملف العناوين <ctype.h> ومنها:-

1) دالة التبديل الى حرف صغير tolower()

وهي اختصار للعبارة (to lower case) حيث تقوم بتبديل الحرف الكبير الى حرف صغير، وفي حالة الحرف الكبير ترجع الدالة بعدد صحيح والمقابل لنفس الحرف في نظام الشفرة (ASCII) بشكله الصغير ومن تم يجري تبديله الى حرف صغير، أما اذا كان الحرف صغيرا فلا يتم تبديله.

مثال 6-2) يوضح البرنامج التالي كيفية استخدام هذه الدالة.

```
        // Program using tolower function
#include <iostream.h>
#include <ctype.h>
main()
{
        char ch;
        cout<<" Enter a character please ==>: ";
        cin>>ch;
        cout<<endl<<"According to tolower"<<endl;
        char result=(char) tolower(ch);
        cout<<ch<< " is converted to lowercase " <<result<<endl;
        getch();
        return 0;
}
```

في حالة تنفيذ هذا البرنامج، وادخال الحرف M مثلا

```
Enter a character please ==>:M
```

سيكون الناتج هو

```
According to tolower
M is converted to lowercase m
```

2) دالة التبديل الى حرف كبير ()toupper

جاءت هذه الدالة اختصارا للعبارة (to upper case) ومهمتها تبديل الحرف الصغير الى حرف كبير، حيث ترجع الدالة بعدد صحيح مقابل للحرف المدخل ومن ثم يجري تبديله اذا كان صغيرا الى حرف كبير، والا فسيبقي الحرف كما هو.

مثال 6-3) اذا ما استبدلت الدالة tolower بالدالة toupper مع بعض التعديلات في جملة cout بالبرنامج السابق يكون ناتج تنفيذه هو الآتي:

```
Enter a character please ==>:a
According to toupper :
a is converted to uppercase A
```

4.6) دوال معالجة السلسلة الحرفية String Manipulation Functions

قبل الدخول في شرح الدوال التي تقوم بمعالجة السلاسل الحرفية ، يجدر بنا التنويه باستخدام ملف العنواين <string.h> الذي يحتوى على تعريفات لهذه الدوال.

1) دالة القياس ()strlen

هذه الدالة هي اختصار للعبارة (string length)، حيث مهمتها الرجوع بقيمة عددية صحيحة تمثل طول أي سلسلة حرفية مع عدم حساب رمز نهاية هذه السلسلة وتأخذ الشكل العام التالي:-

```
strlen(string1)
```

حيث string1 متغير من النوع السلسلة الحرفية.

خذ مثلا العبارة

```
length=strlen("HOW ARE YOU");
```

هنا يتم اسناد طول هذه السلسلة المحصورة بين علامتي التنصيص والتي تحتوي على 11 رمزا (9 حروف ومسافتين خاليتين) للمتغير length .

2) دالة الوصل strcat()

وهي اختصار للعبارة (string concatenation) ومهمتها وصل عدد من المتغيرات الحرفية بعضها ببعض وشكلها العام هو:-

```
strcat(string1,string2)
```

حيث string1 و string2 متغيران من النوع الحرفي ، وهي تعني وصل السلسلة الحرفية بالمتغير string2 عند نهاية السلسلة الحرفية بالمتغير string1 ، وعليه يجب حجز الطول المناسب للمتغير string1 لأنه بعد عملية الوصل سوف يحتوى على طول السلسلتين معا.

3) دالة الوصل حتى n حرف strncat()

هذه الدالة لها الشكل التالي:-

```
strncat(string1,string2,n)
```

حيث تقوم هذه الدالة بأضافة n من الحروف بداية من أقصي يسار السلسلة الحرفية بالمتغير string2 إلى نهاية السلسلة الحرفية (أقصى اليمين) بالمتغير string1.

مثال 4-6) البرنامج التالي يوضح كيفية استخدام هاتين الدالتين.

```
        // Program Using strcat() & strncat() Functions
#include <iostream.h>
#include <string.h>
main()
{
      char *str1,*str2,*str3="";
      cout<<"Type string1 please ==>:";
      cin.getline(str1,80);
      cout<<"Type string2 please ==>:";
      cin.getline(str2,80);
      strcat(str1,str2);
      cout<<"\nResult of strcat(str1,str2) =====>:"<<str1;
      strncat(str3,str1,5);
      cout<<"\nResult of strncat(str3,str1,5) ==>:"<<str3;
      strcat(str3,str1);
      cout<<"\nResult of strcat(str3,str1) =====>:"<<str3;
      getch();
      return 0;
}
```

بعد التنفيذ وادخال البيانات المناسبة على النحو التالي :-

```
Type string1 please ==>:Nice to
Type string2 please ==>:meet you
```

سيكون الجواب كالآتي :-

```
Result of strcat(str1,str2) =====>:Nice to meet you
Result of strncat(str3,str1,5) =====>:Nice
Result of strcat(str3,str1) =====>:Nice Nice to meet you
```

حيث تم وصل الحروف بالمتغير الثاني str2 في آخر حروف المتغير الاول str1 ونتج عن ذلك السلسلة الحرفية "Nice to meet you" تم اخذ الحروف الخمسة الأولى (اربعة حروف وفراغ واحد) الموجودة بالمتغير الاول وهي " Nice"وحفظها في المتغير str3 ، اخيرا تم وصل محتويات المتغير الاول بمحتويات المتغير الثالث ومن ثم حفظ الناتج في المتغير str3 .

4) دالة النسخ strcpy()

هذه الدالة هي اختصار للعبارة (string copy) وتأخذ الشكل العام التالي :-

```
strcpy(string1,string2);
```

وتعني نسخ السلسلة الحرفيه المخزنه في المتغير string2 الى المتغير الحرفي string1 بما فيها رمز نهاية السلسلة ('\0') وفي هذه الحالة يفقد المتغير string1 قيمته القديمة التي قبل النسخ.

فمثلا العبارة

```
strcpy(str,"This is a test ");
```

ينتج عنها نسخ السلسلة الحرفية " This is a test" وتخزينها في المتغير str الذي يجب أن يكون مناسبا من حيث النوع والطول.

5) دالة نسخ n حرف strncpy()

هذه الدالة شكلها هو الآتي: -

```
strncpy(string1,string2,n);
```

أي نسخ n من الحروف بالضبط من المتغير string2 بحيث تنتهي عملية النسخ من المتغير string2 عند الحصول على الرمز ('\0') في حالة ما اذا كان المتغير string2 يتضمن أقل من n حرفا، واذا كان المتغير string2 يحتوي علىn من الحروف أو أكثر، عندها ينسخ n حرفا فقط وقد لا تنتهي السلسلة الحرفيه المخزنه في المتغير string1 بالرمز ('\0').

مثال 6-5) البرنامج التالي مهمته توضيح كيفية استخدام الدالتين السابقتين.

```
        // Program Using strcpy() & strncpy() Functions
#include <iostream.h>
#include <string.h>
main()
{
    char A[80],B[80],C[80];
    cout<<"Get string of A ==>: ";
    cin.getline(A,80);
    cout<< "The string you typed ==>: "<<A<<endl;
    strcpy(B,A);
    cout<< "The string using strcpy(B,A) ==>: "<< B<<endl;
    strncpy(C,A,9);
    C[10]='\0';
    cout<< "The string using strncpy(C,A,13) ==>: "<<C<<endl;
    getch();
    return 0;
}
```

اذا ما نفذ هذا البرنامج واعطيت البيانات المناسبة التالية :-

```
Get string of A ==>:Good luck all of you
```

عندها يرد البرنامج بالاجابة المشابهة للآتي :-

```
The string you typed ==>:Good luck all of you
The string using strcpy(B,A) ==>:Good luck all of you
The string using strncpy(C,A,13) ==>:Good luck all
```

هنا وفي هذا البرنامج تم نقل محتويات المتغير A وتخزينها بالمتغير B ، اما عملية النسخ الثانية تعني نقل الحروف الثلاثة عشر الاولي من المتغير الحرفي A ووضعها في المتغير الحرفي الثالث C .

أما اذا ما غيرنا القيمة 13 لتصبح 9 مثلا ، فالناتج يكون كالآتي:-

```
The string using strncpy(C,A,9) ==>:Good luck
```

6) دالة المقارنة strcmp()

وهي اختصار للعبارة (string compare) وتستخدم لمقارنة متغيرين من النوع الحرفي حيث ينتج عنها قيمة صحيحة وشكلها العام هو :

```
strcmp(string1,string2);
```

فالدالة strcmp() تقارن بين السلسلتين في المتغيرين string1 و string2 من حيث ترتيبهما في النظام آسكي (ASCII) وترجع :-

1) بعدد أكبر من الصفر اذا كان ترتيب string1 أكبر من string2.
2) بعدد يساوي صفرا اذا كانت string1 تكافيء string2.
3) بعدد أصغر من الصفر اذا كان ترتيب string1 أقل من string2.

ملاحظة: الدالة strcmp() ترجع بقيمة عددية أقل من الصفر، وذلك عند مقارنة السلسلة التي تبدأ بالحرف 'A' والسلسلة الاخرى التي تبدأ بالحرف 'B'، والسبب لان الحرف 'A' يأتي قبل الحرف 'B' من حيث الترتيب في نظام آسكي (ASCII)، وهذا ينطبق أيضا اذا تمت مقارنة الكلمة "cat" مع الكلمة "fat" لأن حرف 'c' أصغر من حرف 'f'، ويجب الأخذ في الاعتبار أن الحروف الكبيرة تكون أصغر من الحروف الصغيرة كما يبينه جدول نظام آسكي (ASCII) بالملحق رقم (2).

مثال 6-6) البرنامج التالي يبين استخدام دالة المقارنة strcmp() لسلسلتين من نفس الطول.

```
            // Program Using strcmp() Function
#include <iostream.h>
#include <string.h>
main()
{
```

```
    char str1[80],str2[80],str3[80];
    int result;
    cout<< "Type string1 please ==>:";
    cin.getline(str1,80);
    cout<< "Type string2 please ==>:";
    cin.getline(str2,80);
    result=strcmp(str1,str2);
    cout<< "\nResult of strcmp(str1,str2) =====>:" << result;
    getch();
    return 0;
}
```

عند تنفيذ البرنامج السابق وادخال البيانات المناسبة

Type string1.please ==>:ABCDIC CODE

Type string2 please ==> :abcdic code

سيكون الناتج كالآتي:-

Result of strcmp(str1,str2) =====>: -32

نلاحظ هنا أن الدالة رجعت بقيمة سالبة (32-)، لأن السلسة الاولي عند المقارنة هي أقل من السلسلة الثانية والسبب هو أن الحروف الكبيرة أصغر من الحروف الصغيرة.

فاذا ما نفذ نفس البرنامج ولكن ببيانات تختلف عن السابقة كالآتي:-

Type string1 please ==>:abcdic code

Type string2 please ==>:abcdic code

سيعطي السطر التالي :-

Result of strcmp(str1,str2) =====>: 0

فالقيمة المرجعة عن طريق الدالة هي صفر، أي أن السلسلتين متطابقتان.

7) دالة مقارنة n حرف strncmp()

هذه الدالة تأخذ الشكل التالي:-

strncmp(string1,string2,n);

فهي تقارن قيمة المتغير string1 حتي n حرف من السلسلة string2.

مثال 6-7) يمكن توضيح عمل هذه الدالة عن طريق البرنامج التالي:-

```
                    // Program Using strncmp() Function
#include <iostream.h>
#include <string.h>
main()
{
      char str1[80],str2[80];
      int result;
      cout<< "Type string1 please ==>:";
      cin.getline(str1,80);
      cout<< "Type string2 please ==>:";
      cin.getline(str2,80);
      result=strncmp(str1,str2,4);
      cout<< "\nResult of strcmp(str1,str2,4) =====>:" << result;
      getch();
      return 0;
}
```

عند تنفيذ البرنامج وادخال البيانات المناسبة

```
Type string1 please ==>:HARD WORK
Type string2 please ==>:HARD WARE
```

سيعطي النتائج التالية:-

```
Result of strncmp(str1,str2,4) =====>: 0
```

تمت مقارنـة الأربعـة حـروف الأولـى مـن السلسـلة الحرفيـة الاولـى أي "HARD"، مع الاربعة حروف الاولي من السلسلة الحرفية الثانيه وهـي نفس الكلمة "HARD" نتج عن ذلك تطابق تام، والرجوع بالقيمة صفر.

ماذا يحدث عند ادخال السلسلتين الآتيتين لنفس البرنامج ؟:-

السلسلة الأولي "haRD WORK "

السلسلة الثانية "hARD WARE"

النتيجة ستكون مشابهة للآتي :-

Result of strncmp(str1,str2,4) =====>: 32

فالدالة هنا أعطت قيمة موجبة ، وهي تعني مقارنة الحروف "haRD" مـع الحـروف "hARD" حرفـا حرفـا حيث الحـرف 'h' هـو نفس الحـرف فـي السلسلتين ولكن الفرق بين الحرفين في الخانة الثانية ، حيث الحرف 'a' أكـبر من الحرف 'A' وعليه كانت النتيجة موجبة.

5.6) دوال تحويل السلسلة String Conversion Functions

توجد بعض الدوال في لغة سى++ مهمتها تحويل السلسلة التـي بهـا ارقـام الى قيم صحيحة أو حقيقية، بشرط استخدام ملف العناوين <stdlib.h> مع هـذه الدوال، والجدول التالي يبين البعض منها:-

الدالة	معناها
atof()	تحويل السلسلة الحرفية الى عدد حقيقي مضاعف double
atoi()	تحويل السلسلة الحرفية الى عدد صحيح int
atol()	تحويل السلسلة الحرفية الى عدد طويل long
strtod()	تحويل السلسلة الحرفية الى عدد حقيقي مضاعف double
strtol()	تحويل السلسلة الحرفية الى عدد طويل long
strtoul()	تحويل السلسلة الحرفية الى عدد طويل بدون اشارة unsigned long

مثال 6-8) البرنامج التالي يوضح كيفية استخدام أحد هذه الدوال atof().

```
                    //     Program using atof Function
#include <iostream.h>
#include <stdlib.h>
main()
{
        char str1[20],str2[20];
        cout<< "\nEnter string1 ==>:";
        cin.getline(str1,20);
        cout<< "Enter string2 ==>:";
        cin.getline(str2,20);
        double result=atof(str1)/atof(str2);
        cout << "\nThe result of " << str1 << " / " << str2
              << " Using atof Function = " << result;
        return 0;
}
```

عند التنفيذ ستظهر رسالتان على شاشة العرض وعليه يمكن ادخال قيمة سلسلتين من النوع الرقمي كالآتي:-

```
Enter string1 ==>:19
Enter string2 ==>:3
```

هنا سيتم تحويل كل من السلسلتين الـى قيم عددية من النوع المضاعف حيث تمت عملية القسمة والناتج هو المشابه للآتي :-

The result of 19/3 Using atof Function =6.333333

مثال 6-9) في البرنامج التالي يجري توضيح استخدام الدالة () atol .

```
                    // Program using atol Function
#include <iostream.h>
#include <stdlib.h>
main()
{
        char str1[20],str2[20];
        cout<< "\nEnter string1 ==>:";
        cin.getline(str1,20);
        cout<< "Enter string2 ==>:";
        cin.getline(str2,20);
        long result=atol(str1)*atol(str2);
        cout << "\nThe result of " << str1 << " * " << str2
              << " Using atol Function = " << result;
        return 0;
}
```

فيما يلي توضيح للمدخلات والنتائج التي نحصل عليها عند تنفيذ هذا البرنامج:-

Enter string1 ==>:55555

Enter string1 ==>:10000

The result of 55555 * 10000 Using atol Function =555550000

مثال 6-10) في البرنامج التالي يجرى توضيح استخدام الدالة () strtod.

```
                    // Program using strtod Function
#include <iostream.h>
#include <stdlib.h>
#include <string.h>
```

```
main()
{
      char str1[80],str2[80];
      char *ptr;
      cout<< "\nEnter string1 ==>:";
      cin.getline(str1,80);
      double result=strtod(str1,&ptr);
      cout << "\nBy using strtod Function : "
           << "\nThe double value string ==> " << result
           << "\nWhile remaining string ==> " << ptr;
      getch();
      return 0;
}
```

بعد تنفيذ البرنامج وادخال السلسلة

Enter string1 ==>:100.99 dinar in my pocket

سيتم اضهار النتائج على النحو التالي:-

By using strtod Function :

The double value string ==> 100.99

While remaining string ==> dinar in my pocket

في هذا البرنامج استخدمت الدالة strtod التي لها معاملان، المعامل الاول هو السلسلة الحرفية str1 حيث تم تحويل القيمة الحقيقية في هذه السلسلة الى قيمة مضاعفة، اما المعامل الثاني فهو المؤشر ptr الذي يؤشر الى الموقع الاول من السلسلة المدخلة وهو الحرف d بعد التحويل، عندها تم اسناد بقية السلسلة وهي dinar in pocket الى المتغير ptr.

مثال 6-11) في البرنامج التالي يجرى توضيح استخدام الدالة strtol ().

```
                  // Program using strtol Function
#include <iostream.h>
#include <stdlib.h>
#include <string.h>
```

```
main()
{
        char str1[20],str2[20];
        char *ptr;
        cout<< "\nEnter string1 ==>:";
        cin.getline(str1,20);
        long result=strtol(str1,&ptr,0);
        cout << "\nBy using strtol Function : "
             << "\nThe long value string ==> " << result
             << "\nWhile remaining string ==> " << ptr;
        getch();
        return 0;
}
```

الدالة strtol() وظيفتها تحويل القيمة الصحيحة بالسلسلة الى قيمة من النوع الطويل وهي تحتوي على ثلاثـة معـاملات، الاول السلسلة المـراد معالجتهـا، الثاني هو مؤشر الى الموقع الذي به الحرف الاول بعد التحويل اما المعـامل الثالث 0 فيشـير الـى امكانيـة تحويـل القيمـة المحولـه الـى النظـام الثمـاني أو العشري أو الستة عشرى، وفيما يلي البيانات المدخلة مع المخرجات في حالة تنفيذ هذا البرنامج.

```
Enter string1 ==>:12345 is five digits
By using strtol Function :
The long value string ==> 12345
While remaining string ==> is five digits
```

6.6) تمارين Exercises

(1) اكتب برنامجا لطباعة الرسالة المناسبة توضح هل الحرف المدخل عن طريق لوحة المفاتيح رقم في النظام الستة عشرى أم علامة ترقيم.

(2) أكتب برنامجا لادخال عدد من الجمل لا تتعدي الواحدة منها 20 حرفا ، المطلوب طباعة أقصر جملة ثم أدخالها.

(3) اكتب برنامجا كاملا يقرأ حرفا أبجديا ومن ثم يطبع موقعه في الحروف الابجدية.

(4) اكتب برنامجا مهمته قراءة سلسلة حرفية ثم يطبع yes اذا كانت السلسلة تقرأ من اليمين الى اليسار أو بالعكس أو يطبع no اذا كان غير ذلك ، فمثلا اذا كانت السلسلة "i am ma i" عندها تطبع الكلمة yes .

(5) أكتب برنامجا لادخال سلسلة حرفية string ثم أحسب الآتي :-

* عدد الكلمات التي تبدأ بالحرف B.
* عدد الحروف الصغيرة في هذه السلسة.
* عدد وجود الحرف N في هذه السلسلة.

(6) أكتب برنامجا يستقبل سلسلتين str2,str1 ثم يطبع Positive اذا كان str1 أكبر من str2 والكلمة Negative اذا كان str1 أصغر من str2 والكلمة Zero اذا كانت السلسلتان متطابقتين.

(7) اكتب برنامجا مهمته قراءة سلسلة حرفية طولها 15 حرفا اولا ثم طباعة yes اذا كان الرمز المدخل من لوحة المفاتيح موجودا في هذه السلسلة مع عدد مرات ظهوره، وطباعة no اذا كان غير ذلك.

(8) المطلوب كتابة برنامج مهمته استقبال سلسلة حرفية ونسخ هذه السلسلة في متغير آخر فيما عدا الحرفين B,N.

(9) بافتراض أننا اعطينا الأمر:-

```
char str1[20]="12345",str2[20]="ABCDEFGH";
```

المطلوب ايجاد ما يطبع بالفقرات التالية:-

(a)

```
strncpy(str1,str2,4);
cout<<"The string1 ==>: "<<str1;
```

(b)

```
int result=strcmp(str1,str2);
cout<<endl<< "Result ==>:" << result;
```

(c)

```
strcat(str1,strncpy(str2,str1,3));
cout<<"The str1 ==> "<<str1;
```

(10) اكتب برنامجا كاملا يقرأ سطرا به عدد محدد من الرموز ينتهي بالرمز (?) ثم يقوم بالآتي:-

* طباعة وحساب عدد مرات تكرار الحروف O,T,N .
* طباعة وحساب عدد مرات تكرار الارقام 5,3,1 .
* طباعة السطر بعد استبدال الحروف في الفقرة الاولى بالمقابل لها في الفقرة الثانية.

(11) أكتب برنامجا لادخال عدد N من أرقام الهواتف، المطلوب اضافة الرقمين 33 من الناحية اليسري للرقم الذي يبدأ بالرقم 3، وأضافة الرقمين 44 من الناحية اليسري للرقم الذي يبدأ بالرقم 4 .

(12) اكتب برنامجا لادخال سلسلة حرفية عن طريق لوحة المفاتيح ثم قم بتحويل الحروف الكبيرة الى حروف صغيرة بهذه السلسلة.

(13) اوجد ناتج تنفيذ البرنامج التالي:-

```
#include<iostream.h>
#include <string.h>
main()
{
	char *str1="aabbbb",*str2="aaabbb",*str3="ccc";
	if(strncmp(str2,str1,3)>0)
		cout<<"str2>str1"<<endl;
	else
		cout<<"str1>str2"<<endl;
	if(strncmp(str2,str3,3)>0)
		cout<<"str2>str3"<<endl;
	else
		cout<<"str3>str2"<<endl;
}
```

(14) اكتب برنامجا لقراءة سلسلة حرفية لا يزيد طولها على 15 رمزا ثم اطبع هذه السلسلة عدد مرات طولها مع الغاء الرمز الاخير من هذه السلسلة في كل مرة وحتى النهاية ، فمثلا السلسلة STRING ستطبع بالشكل التالي:-

```
STRING
STRIN
STRI
STR
ST
S
```

الفصل السابع: الدوال

من خلال استعراضنا للبرامج التي كتبناها سابقا، يتضح أنها نفذت ككتلة واحدة غير مجزأة، الا أن هذا الوضع قد لا يكون مناسبا في بعض الأحوال، فالبرنامج الذي يجري تنفيذه قد يكون كبيرا ويحتوي على تفاصيل وتفريعات كثيرة تجعل من الصعب التعامل معه ومتابعته وتصحيح ما قد يقع فيه من أخطاء.

عليه، يصبح من المستحسن وأحيانا من الضروري تقسيم مثل هذا البرنامج الى عدة أجزاء يعرف كل منها بالدالة الفرعية، بحيث تتولى كل دالة القيام بمهمة معينة يسهل اختبارها وربطها مع الدالة الرئيسية main ().

1.7) تعريف الدالة Function Definition

الدالة عبارة عن برنامج فرعي يستطيع المبرمج كتابتها وبالتالي استخدامها عن طريق استدعائها بواسطة الدالة الرئيسية أو أي دالة فرعية أخرى، وفي لغة سى++ تستخدام العينة (Prototype) مع الدالة حتي يمكن التغلب على اختلاف البيانات المرسلة من والى الدالة، والشكل العام للدالة هو:

```
                    // Function Prototype
Type_Returned  Function_Name(Parameter_List);
                    //  Function Definition
Type_Returned Function_Name(Parameter_List);
{
        .
        . //        function body
        .
}
```

حيث:

Type_Returned يمثل نوع القيمة المرجعة من الدالة الى نقطة الاستدعاء.

Function_Name يمثل اسم الدالة.

Parameter_List قائمة بالادلة أو المعاملات لاستقبال وارجاع البيانات.

function body جسم الدالة ويمكن ان يحتوي على بعض الاعلانات عن المتغيرات الخاصة بالدالة بالاضافة إلى جملة أو مجموعة من الجمل التنفيذية. وقد تنتهي الدالة بالامر return وذلك لارجاع القيمة الى نقطة استدعاء الدالة.

ملاحظة: عند استخدام الدالة يجب مراعاة الآتي :-

1) من الضروري أن يتبع اسم الدالة قوسان مثل sample () وقد يحتوي القوسان على ادلة أو معاملات (parameters) وتسمي بالادلة الصورية وتستخدم عادة لتبادل البيانات بين الدالة الرئيسية والدالة الفرعية، أو قد لا يحتويان على أي شيء (فارغة).

2) اسم الدالة لا يتبعها الفاصلة المنقوطة (;) فالدلة func1

```
double func1(value)
```

تضم دليلا صورياً واحدا هو value أما عند استدعائها فيجب ان تضم دليلا فعليا أو حقيقيا واحدا مقابلا للدليل value وهو num مع انهائها بالفاصلة المنقوطة.

```
result=func1(num) ;
```

3) ينبغي الاعلان عن نوع الدالة مسبقا اذا كانت ترجع بقيمة من النوع غير الصحيح int.

4) قد تأتي الدالة الفرعية قبل أو بعد الدالة الرئيسية main().

5) عند تمرير البيانات من الدالة الرئيسية الى أي دالة فرعية أو بالعكس يجب ان تكون الادلة أو المعاملات المتصلة بينها متوافقة من حيث العدد والنوع.

2.7) الدالة الفارغة void Function

قد يكون للدالة مهمة معينة تؤديها بدون ارجاع قيمة عند انتهائها ونقول ان الدالة خالية بدون معاملات.

مثال 7-1) الدالة الفارغة التي تحت اسم message قد تأخذ التعريف التالي:-

```
void message (void);
```

اما وقت استدعائها فقد تأخذ الشكل التالي :-

```
message ();
```

مثال 7-2) البرنامج التالي يوضح استخدام الدالة الفارغة void

```
            // Functions that take no arguments
#include<iostream.h>
void message (void); // Function prototype
main()
{
      cout<<endl<<"First call"<<endl;
      message ();        // First call
      cout<<endl<<"Second call"<<endl;
      message ();        // Second call
      getch();
      return 0;
}
void message ()
{
      cout<<"\tHi my friend welcome to computer center ";
}
```

في هذا البرنـامج تم الاعـلان عـن نـوع الدالـة message قبل بدايـة الدالـة الرئيسية باسـتخدام العينـة (Prototype)، تـلا ذلك اسـتدعاء الدالـة مرتين عن طريق اسمها ()message مـع ملاحظـة انتهـاء الدالـة بالفاصلـة المنقوطـة عنـد الاعلان عنها واستدعائها.

اما بخصوص الدالة الفرعية message فيجب مراعاة ما يلي:-

1) أن يكون هناك توافق بين نوع الدالة والاسم المعلن عنـه فـى الدالـه الرئيسية.
2) أن تكون الدالة متبوعة بالقوسين () الدالين على خلوها من أيـة قيمـة كما يجب عدم انتهائها بالفاصلة المنقوطة.
3) أن يأتي بعدها جسم الدالـة المحصـور بيـن القوسـين { } وفيـه تـم طباعة الرسالة المناسبة.
4) عدم انتهاء الدالة بالامر returnلعدم رجوع الدالة بأية قيمة .

وعلى العموم فعند اتمام التنفيذ سوف تظهر لنا النتائج المشابهة للآتي:-

```
First call
Hi my friend welcome to computer center
Second call
Hi my friend welcome to computer center
```

3.7) المتغيرات المحلية Local Variables

هي متغيرات يطلق عليها بعض الاحيـان المتغيرات الداخليـة ، حيث يتـم الاعلان عنها فى حدود نطاق الدالة، وعليه لا يمكن التعرف عليهـا فـى أيـة دالة أخرى حتي ولو كانت تحمل نفس الاسم.

مثال 7-3) البرنامج الموالي وظيفتـه استدعاء دالـة لايجـاد مساحة المستطيل rectangle وهي القاعدة في الارتفاع.

```
#include<iostream.h>
#include <stdlib.h>
                         // Functions prototype
void rectangle (void);
main()
{
        rectangle();
        return 0;

void rectangle()
{
        float L,W;          // local variables
        cout <<"\nEnter length and width of rectangle ==>:";
        cin>>L>>W;
        float area=L*W;
        cout <<"The area of rectangle is "<<area;
}
```

عند تنفيذ البرنامج يتم أستدعاء الدالة rectangle والتي هي بدون معاملات حيث تم الاعلان عن المتغيرات W , L (طول وعرض اضلاع المستطيل) من النوع الحقيقي بعد القوس المفتوح مباشرة، وعليه تعتبر هذه المتغيرات محلية أي تخص هذه الدالة فقط ثم تطبع الرسالة الدالـة على ادخال القيمتين وبالتالي حساب مساحة المستطيل وحفظها في المتغير المحلي Area وطباعتها والرجوع إلى الدالة الرئيسية ونهاية البرنامج.

4.7) التمرير بالقيمة Passing by Value

تخول لنا هذه اللغة تمرير قيم من الدالة الرئيسية الى أي دالة أخرى تابعة لها عن طريق معاملاتها، وبالتالي الرجوع بقيمة واحدة وقت انتهائها.

مثال 4-7) البرنامج التالي مهمته قراءة اربعة متغيرات من النوع الحقيقي مع مناداة دالة واحدة وظيفتها ايجاد اكبر قيمة من هذه المتغيرات الاربعة.

```
            // Get maximum value using function
#include<iostream.h>
                    // Functions prototype
void nothing(void);
float larger(float,float);
main()
{
    float num1,num2,num3,num4;
    nothing();
    cin>>num1>>num2>>num3>>num4;
    float temp1=larger(num1,num2);
    cout <<endl<<"the larger of the first"
         <<" two numbers is "<<temp1;
    float temp2=larger(num3,num4);
    cout <<endl<<"the larger of the"
         <<" secand two numbers is "
         <<temp2; float max=larger(temp1,temp2);
    cout<<endl<<"the largest number is "<<max;
    getch();
    return 0;
 }
                         //    print the message
void nothing()
{
     cout <<endl<<"find the largest of four numbers "<<endl
          <<"------------------------------"<<endl
          <<"Type  four data items ==>: ";
}
                         //    return the maximum number

float larger(float x,float y)
{
     if(x>y)
          return x;
     else
          return y;
}
```

ما سيطبعه البرنامج عند تنفيذه هو الناتج المشابه للآتي:-

```
find the largest of four numbers
-------------------------------
 Type  four data items ==>: 9.999 5.555 0.044 0.444


The larger of the first two numbers is 9.999
The larger of the second two numbers is 0.444
The largest number is 9.999
```

يبدأ البرنامج الرئيسي بالاعلان عن الدالة الاولى

```
void nothing(void);
```

وتعني أنها دالة بلا معاملات والثانية

```
float larger(float,float);
```

وتعني أنها دالة مررت لها قيمتان من النوع الحقيقي ويطلق عليه استدعاء الدالة بالقيمة (call by value) عن طريق معاملاتها الصورية وترجع بقيمة من النوع الحقيقي.

بعدها يتم استدعاء الدالــة nothing() والخاليــة مـن أي معـاملات ومهمتهـا طباعة الرسالة التالية:-

```
find the largest of four numbers
-----------------------------
Type  four data items ==>:
```

ثم يجري الرجوع الى الجملة الموالية لجملـة الاسـتدعاء وهـي جملـة قنـاة ادخـال البيانـات cin وذلك لادخـال اربعة قيم حقيقيـة، بعـد الانتهـاء مـن ادخـال القيم المطلوبة، تستدعى الدالة larger عن طريق الجملة

```
float temp1=larger(num1,num2);
```

حيث تمرر قيمة كل من الدليلين الحقيقيين num2,num1 الى الدليلين الصوريين y,x على التوالي فى الدالة larger وبالتالي الرجوع بأكبر قيمة وتخصيصها للمتغير temp1 ، ويتكرر نفس الشيء كلما استدعيت الدالة larger ولكن بقيم مختلفة.

أما فيما يخص شكل الدالة larger فقد بدأت بالاعلان عن نوعها كمايلي:-

float larger (float x,float y)

ويدل ذلك على انها دالة من النوع الحقيقي وتحتوي على معاملين y , x تم الاعلان عنهما مباشرة داخل القوسين() الخاصين باسم الدالة.

اما جسم هذه الدالة فيحتوي على جملة if التي مهمتها المقارنة بين قيمتي المتغيرين y,x والرجوع بأكبرهما عن طريق جملة return.

ميزة كبيرة في استخدام الدالة تتمثل في تجنب تكرار جملة المقارنة لاكثر من مرة في البرنامج الرئيسي.

يمكن اشهار دليل الدالة باستخدام الثابت const لاستقبال قيمة معينه من نقطة الاستدعاء ولكن بشرط ألا يتغير هذا الثابت عند استعماله في الدالة بالقرأة أو الزيادة أو النقصان.

مثال 7-5) ما هو ناتج البرنامج التالي ؟.

```
#include <iostream.h>
main()
{
        float func(float); // Function prototype
        int value;
        float ansr=func(value);
        cout<<"ansr="<<ansr;
        getch();
```

```
        return 0;
}

float func(float n1)
{
        n1=25;
        return n1;
}
```

هنا يتم استدعاء الدالة func والتـي مهمتهـا ارجـاع القيمـة 25 عـن طريـق المتغير الصورى n1 الى نقطة الاستدعاء وتخصيص هذه القيمة للمتغير ansr ليطبع البرنامج الآتي:-

ansr=25

اما اذا تغيرت عينة الدالة لتصبح

```
float func(const float);
```

وتعريفها ليصبح

```
float func(const float n1)
```

في البرنامج السابق باستخدام الثابت const لدليل الدالة n1، فسوف يظهر لنا عند التنفيذ رسالة الخطأ التالية:-

Connot modify a const object

التي تعني لايمكن التغيير في قيمة الثابت n1 باسناد القيمة 25 اليه.

5.7) المتغيرات العامة Global Variables

المتغيرات العامة أو الشاملة هي التي تكون معروفة للدالة الرئيسية main() وبقية الدوال الفرعية الاخرى، ويتم الاعلان عنها قبل بداية الدالة الرئيسية ولا يجوز الاعلان عنها لكثر من مرة.

مثال 7-6) البرنامج التالي يبين الفرق بين المتغير المحلي والخارجي.

```
#include<iostream.h>
int outer;                    // External variable
main()
{
        void call_it(int); // Function prototype
        outer=5;
        int inner=5;
        cout<<endl<<"The output in the first call ";
        call_it(inner);
        cout <<endl<<"outer= "<<outer<<"  inner= "<<inner
             <<endl<<"The output in the second call ";
        call_it(inner);
        cout <<endl<<"outer= "<<outer<<"  inner= "<<inner;
        getch();
        return 0;
}

void call_it(int m)
{
        cout<<endl<<"-----------------------------";
        outer*=outer;
        m*=m;
        return;
}
```

في هذا البرنامج تم الاعلان عن المتغير outer قبل بداية الدالة الرئيسية main() وبالتالي فهو يعتبر متغيراً خارجياً أو شاملاً، اما المتغير inner والذي تم الاعلان عنه في نطاق الدالة الرئيسية يعتبر متغيرا داخليا أو محليا وهو معروف للدالة الرئيسية فقط.

عند استدعاء الدالة call_it() في المرة الاولى مررت القيمة 5 عن طريق المتغير inner من خلال الدالة الرئيسية الى المتغير المحلي m بالدالة call_it() وتمت مضاعفة قيمة كل من المتغيرين outer , m الذين لهما نفس القيمة وهي 5 وليصبح كل واحد منهما يساوي 25، ولكن عند الرجوع الى نقطة

الاستدعاء في الدالة الرئيسية، لم تتغير قيمة المتغير inner فبقيت قيمته كما هي 5 بينما تم تغيير قيمة المتغير الخارجي outer لتصبح 25 ، وعند استدعاء نفس الدالة للمرة الثانية يتم مضاعفة قيمة المتغير الخارجي outer وهي القيمة الاخيرة 25 لتصبح تساوي 625 ، بينما تبقي قيمة المتغير المحلي inner كما هي عليه في السابق، وفيما يلي النتائج الكلية لهذا البرنامج وقت تنفيذه:-

```
The output in the first call
-----------------------------
outer=25    inner=5
The output in the second call
-----------------------------
outer=625    inner=5
```

ماذا يحدث اذا تم اعادة الاعلان عن المتغير الخارجي outer بالدالة الفرعية call_it مرة اخرى؟.

مثال 7-7) للاجابة عن هذا السؤال نقوم بتغيير الدالة call_it في المثال السابق لتصبح

```
void call_it(int m)
{
        int outer;
        cout<<endl<<"-----------------------------";
        outer*=outer;
        m*=m;
        return;
}
```

اذا تم الاعلان عن المتغير outer من النوع الصحيح int مرة أخرى، عندها يصبح متغيرا محليا خاصا بهذه الدالة مثله مثل المتغير المحلي m ولا

يتم ارجاع قيمته الى الدالة الرئيسية، وبالتالي يكون ناتج تنفيذ هـذه الدالـة مـع الدالة الرئيسية القيمة 5 للمتغيرين outer و inner في الحالتين.

6.7) المتغيرات الساكنة Static Variables

هذا النوع من المتغيرات سميت بالساكنة لانها تحتفظ بقيمتها المخزنة بعد الانتهاء من تنفيذ الدالة طوال تنفيذ البرنامج.

الشكل العام

```
static type variable;
```

مثال 7-8) البرنامج التالي مهمته توضيح استخدام المتغير الساكن.

```
#include<iostream.h>
int use_static_fun (void);
main()
{
        for(int i=1; i<10; i++)
              count << i <<"*"<< i << use_static_fun( ) <<endl;
        return 0;
}
int use_static_fun (void)
{
        static int counter=0;
        counter ++;
        return counter * counter;
}
```

تم الاعلان عن المتغير counter على أنه متغيرساكن ومن النوع الصحيح كعداد في الدالة use_static_fun وفي كل مرة تستدعي هذه الدالة يتم اضافة القيمة 1 الى القيمة السابقة التي احتفظ بها المتغير counter والرجوع بمربع هذه القيمة إلى نقطة الاستدعاء.

اما في حالة حذف الكلمة static من امام المتغير counter في البرنامج السابق، عندها يرجع بمربع العدد 1 في كل مرة تستدعي فيها الدالة لان counter يعمل كمتغير من النوع الصحيح العادي.

7.7) الادلة سابقة التعريف Default Parameters

هذه ميزة اخرى اضافتها لغة سي++ الى المبرمج وهي امكانية شحن ادلة أو معاملات الدالة بقيم سابقة التعريف حيث تظهر اما في عينة الدالة (prototype) أو مع عنوان الدالة بين القوسين () فقط .

مثال 7-9) الدالة

```
float default_one (float f=4.5 , int m=25);
```

تم فيها شحن كل من الدليلين m,f بالقيم الابتدائية 4.5 ,25 على التوالي ، وعليه يمكن استدعاء هذه الدالة بالشكل التالي :-

```
result=default_one();
```

وبالتالي فان القيمتين 25, 4.5 تعتبران سابقة التعريف للدليلين m,f يستخدم هذان الدليلان في نطاق الدالة default_one

مثال 7-10) الدالة

```
int default_two (int f , int m=25);
```

هنا الدليل f ليس له قيمة سابقة التعريف، فعند استدعاء الدالة يجب تمرير قيمة عددية ليحل محل الدليل f ، وفي حالة استدعاء هذه الدالة بالامر

```
result=default_two(10,111);
```

سوف يتم تخصيص القيمة 10 للدليل الأول f في حين ان قيمة الدليل الثاني m تبقي 25 كما هي.

ملاحظة: يجب الاخذ في الاعتبار عند تحديد القيم سابقة التعريف لبعض الادلة، أنه لا يجوز تحديد القيمة الابتدائية للدليل الاول ، أي لا يمكن استخدام الدالة default_two بالشكل التالي

```
int default_two (int k=10, int m);
```

مثال 7-11) البرنامج التالي يقوم باستدعاء دالة فرعية ادلتها سابقة التعريف مهمتها حساب حجم الصندوق الذي يساوي الطول في العرض في الارتفاع.

```
                    // This program using default arguments
#include<iostream.h>
int box_volume(int x=1 , int y=1 , int z=1);
main()
{
        int l=8,w=5,h=2;
        int volume=box_volume(l);          // firest call
        cout <<"The volume of a box with length "<<l<<endl
             <<"width 1 and height 1 is =======>: "<<volume<<endl;
        volume=box_volume(l,w);            // second call
        cout <<"The volume of a box with length "<<l<<endl
             <<"width "<<w<<" and height 1 is =======>: "
             <<volume<<endl;
        volume=box_volume(l,w,h);          // thired call
        cout <<"The volume of a box with length "<<l<<endl
             <<"width "<<w<<" and height "<<h<<" is =======>: "
             <<volume<<endl;
        return 0;
}

int box_volume(int length, int width, int height)
{
        return length*width*height;
}
```

تم شحن الادلة z,y,x بالدالة box_volume في عينـة الدالـة بالقيمـة 1 قبل الدخول في الدالة الرئيسية ، وقد استدعيت هذه الدالة ثلاث مرات:

1) المرة الاول ارسال القيمة 8 التي تمثل طول الصندوق الى الدالة وبالتالي يأخذ الدليل الاول length القيمة 8 بدلا من القيمة الابتدائية 1 في حين يأخذ الدليلان width , height القيمة الابتدائية 1 ومن هنا يتم حساب حجم الصندوق وارجاع القيمة 8 الى نقطة الاستدعاء عن طريق الامر

```
return 8*1*1
```

2) في الاستدعاء الثاني مررت القيمتان 5,8 وهما تمثلان طول وعرض الصندوق وخصصتا للدليلين width , hength على التوالي بينما الارتفاع الذي يمثل الدليل height حدد له القيمة 1 ومن هنا يتم الرجوع بحجم الصندوق وهي القيمة 40 من خلال الجملة

```
return 8*5*1
```

3) وفي المرة الثالثة يتم تمرير كل البيانات المعطاة بالدالة الرئيسية وهي القيم 2,5,8 باستخدام الادلة الحقيقية h,w,l الى الدالة الفرعية باستخدام الادلة الصورية height , width , length للحصول على الحجم وهو القيمة 80 الناتجة من جملة الرجوع

```
return 8*5*2
```

الناتج وقت التنفيذ كالاتي:

```
The volume of a box with length 8
      width 1 and height 1 is ======>: 8
The volume of a box with length 8
      width 5 and height 1 is ======>: 40
The volume of a box with length 8
      width 5 and height 2 is ======>: 80
```

8.7) التحميل الزائد للدوال Function Overloading

لغة سي++ توفر للمستخدم فرصة للاستفادة من خاصية أخرى وهي التحميل الزائد للدوال أي الاعلان عن عدد من الدوال يكون لها نفس الاسم ولكن بدلائل مختلفة من حيث النوع والعدد.

اذا طلب منك كتابة دالة تقوم بالرجوع بمربع القيمة المرسلة اليها ، فان تحقيق هذا الطلب يتوقف على نوع البيانات التي تقوم الدالة بمعالجتها أي بمعني أنه اذا كانت كل البيانات المرسلة من النوع الصحيح int فلا توجد أية صعوبة في كتابة مثل هذه الدالة ، وقد تكون على النحو التالي :-

```
int square(int a);
{
    return a*a;
}
```

اما اذا كانت هذه الدالة تتعامل في المرة الاولي مع متغير من النوع الصحيح int وفي الثانية مع متغير من النوع المضاعف double، فما عليك الا باستعمال التحميل الزائد للدوال أي استخدام نفس اسم الدالة مع اختلاف نوع دليلها حسب النوع المطلوب، فالدالتان :

```
int square(int a);
double square(double b);
```

لهما الاسم square ولكن في الاولي دليلها a من النوع الصحيح وقيمتها عند الرجوع من نفس نوع دليلها بينما في الثانية دليلها b وقيمتها من النوع المضاعف، في حين ان الدالتين:

```
int next(int c);
char next(char d);
```

لهما الاسم next فالأولى تستقبل قيمة من النوع الصحيح int وترجع بقيمة من نفس النوع، بينما الثانية تستقبل قيمة من النوع الحرفي char وترجع بقيمة من نفس نوعها.

مثال 7-12) البرنامج التالي يتم فيه استدعاء اربع دوال ، الاولي والثانية مهمتهما ارجاع مربع القيمة المرسلة لها، امـا الدالتـان الثالثـة والرابعـة فمهمتهما الرجوع بالقيم الموالية للقيمة التي مررت لهما.

```
        // This program using overloading functions
#include<iostream.h>
                        // Functions prototype
int square(int a);
double square(double b);
int next(int c);
char next(char d);
main()
{
      int i=4,n=9;
      double f=5.5;
      char ch='A';
      cout<<"The square of integer "<<i<<" = "<<square(i)<<endl;
      cout<<"The square of double " <<f<<" = "<<square(f)<<endl<<endl;
      cout<<"The next character of "<<ch<<" = "<<next(ch)<<endl;
      cout<<"The next number of "<<n<<" = "<<next(n)<<endl;
      return 0;
}

int square(int x)
{
      return x*x;
}

double square(double y)
{
      return y*y;
}
```

```
char next(char z)
{
        return ++z;
}

int next(int w)
{
        return ++w;
}
```

في البرنامج وقع استخدام الدالتين تحت الاسم square حيث استقبلت الاولي القيمة 4 من النوع الصحيح والثانية القيمة 5.5 من النوع المضاعف مع ارجاع كل واحدة منها بمربع القيمة المرسلة اليها، في حين أن الدالتين الواقعتين تحت اسم next قامت أولاهما باستقبال الحرف A وارجاع الحرف الموالي لهذا الحرف وهو الحرف B والثانية باستقبال القيمة الصحيحة 9 والرجوع بالقيمة الموالية لهذه القيمة وهي القيمة 10، وهذا هو ناتج التنفيذ

```
The square of integer  4 = 16
The square of double 5.5 = 30.25

The next character of A = B
The next number of 9 = 10
```

مثال 7-13) المطلوب شرح البرنامج التالي :-

```
#include <iostream.h>
double avg(double n1,double n2,double n3);
double avg(int n1,int n2);

main()
{
        double d1=10.0,d2=9.5,d3=6.0;
        cout <<"The average of "<<d1<<", "<<d2<<" and "
             <<d3<<" is "<< avg(d1,d2,d3)<<endl;
```

```
        int n1=3,n2=4;
        cout <<"The average of "<<n1<<" and "<<n2<<" is "
              << avg(n1,n2)<<endl;
        return 0;
}

double avg(int a,int b)
{
        return ((a+b)/2.0);
}

double avg(double a,double b, double c)
{
        return ((a+b+c)/3.0);
}
```

نلاحظ في هذا البرنامج انه استدعيت الدالة avg في المرة الأولي ومررت لها ثلاث قيم من النوع المضاعف وارجاع متوسط هذه القيم وطباعة الناتج، ثم استدعيت الدالة التي لها نفس الاسم السابق ولكن مررت لها قيمتان من النوع الصحيح وبالتالي ارجاع متوسطهما بالدالة الرئيسية، أي يمكن اطلاق اسم واحد avg على الدالتين مع اختلافهما من حيث النوع وعدد الادلة، وناتج هذا البرنامج هو الآتي :-

```
The average of 10, 9.5 and 6 is 8.5
The average of 3 and 4 is 3.5
```

9.7) دالة المعاودة الذاتية Recursion Function

المعاودة الذاتية هي استدعاء الدالة الفرعية لنفسها عددا من المرات وهو استدعاء متوقف على تحقيق شرط معين ينهي تكرار عملية الاستدعاء ، واذا لم يتحقق هذا الشرط عندها سيتم استدعاء الدالة لنفسها في النقطه المناسبة.

مثال 7-14) البرنامج التالي يقوم باستدعاء دالـة الاعـادة التـي مهمتهـا حسـاب مجموع أول 5 أعداد.

```
#include<iostream.h>
int sum_it(int m);
main()
{
	int n=5;
	cout <<"The following is the sum of"<<endl
		<<"the first 5 positive integer numbers"<<endl;
	int sum=sum_it(n);
	cout<<"  Which is equal to "<<sum;
	getch();
}

			// ***** Compute the sum  *****
int sum_it(int m)
{
	if(m==1)
	{
		cout<<m;
		return m;
	}
	else
		cout<<m<<" + ";
	return(m+sum_it(m-1));
}
```

هنا تمرر قيمـة المتغير n مـن الدالـة الرئيسـية main() الـى معـامل الدالـة الفرعية sum_it() وهو المتغير الصحيح m وبالتـالي اذا كـان الشـرط (m==1) لجملـة if صحيحـا (true) ، تطبـع القيمـة وترجـع إلـى نقطـة الاستدعاء عـن طريق جملة return، اما اذا كان الشرط خاطئا (false) فتنفذ جملة الطباعة ثم يعقبها تنفيذ الجملة التالية :-

```
return(m+sum_it(m-1));
```

وهي تعني اضافة قيمة المتغير m الى قيمة الدالة sum_it() مع طرح القيمة 1 من المتغير m ، وبالتالي تستدعي الدالة نفسها مرة أخرى ، حيث تخصص نتيجة الطرح الى معامل الدالة المتغير m ، وهكذا تستمر عملية الاعادة حتي تصبح قيمة التعبير (m-1) تساوي 1 عندها يتحقق الشرط جملة if ، وبالتالي تضاف القيمة النهائية للمتغير m الى قيمة الدالة ويتم الرجوع بهذه القيمة الى نقطة الاستدعاء عن طريق جملة return، اذا نفذ هذا البرنامج سيكون الناتج كما يلي :-

```
The following is the sum of
the first 5 positive integer numbers
5+4+3+2+1 Which equal to 15
```

10.7) الماكرو Macro

تطرقنا سابقا الى ذكر بعض الموجهات التي تستخدم عند كتابة برنامج بلغة سي++ وهي التي تبدأ بالرمز # مثل #include ، ايضا هناك توجيه جديد آخر هو #define الذي يقوم بانشاء معرفات تسمي الثوابت الرمزية (Symbolic Constants) وبالتالي يجري تعريف البرنامج بإحدى هذه الثوابت التي يتكرر استخدامها اثناء تنفيذ البرنامج لاكثر من مرة .

خذ مثلا الأمر

```
#define SIZE 50
```

يسبب هذا الامر استبدال الاسم SIZE كلما وجد في البرنامج بالقيمه 50، لاحظ عدم انهاء هذا التوجيه بالفاصلة المنقوطة، والثابت الرمزي يمكن كتابته بالحروف الصغيره أو الكبيرة ولا يمكن تغيير قيمته (بالزياده أو النقصان) أثناء تنفيذ البرنامج، فمثلا من الخطأ أن نكتب الجملة

```
SIZE=SIZE+40;
```

مثال 7-15) فيما يلي بعض الامثلة باستخدام جملة define

```
#define PI 3.141519
#define Days_Per_Week 7
#define STRING "This is a string "
#define SLASH "\"
```

يمكن استخدام الماكرو في اغراض اخرى والذي عن طريقه تنفذ جملة أو عدد من الجمل عوضا عن استخدام الدوال الفرعية كما تعودنا سابقاً.

مثال 7-16) لايجاد أكبر قيمة من متغيرين y,x يمكن كتابة ماكرو large على النحو التالي :-

```
#define large(a,b)   (a>b ? a : b)
```

على ان يستدعي هذا الماكرو عن طريق الجملة التالية من خلال جسم الدالة الرئيسية أو الفرعية.

```
result=large(x,y);
```

وينتج عنه ارسال قيمة المتغيرين y,x الى المتغيرين b,a على التوالي بعدها نحصل على أكبر قيمة منهما حيث يتم ارجاعها وتخزينها في المتغير result

الجملة التالية مهمتها ايجاد تربيع أي قيمة صحيحة للمتغير x .

```
#define square(x)   x*x
```

مثال 7-17) المطلوب شرح البرنامج الموالي مع ايجاد ناتج تنفيذه.

```
#include<iostream.h>
#define  message(text)  cout<< #text
#define  result (m)  cout<<m*m
main()
```

```
{
        int x;
        message(Enter your value ==> : );
        cin>>x;
        message(The square of your value is ==>: );
        result(x);
}
```

في بداية هذا البرنامج تم ارسال الرسالة الاولى .

Enter your value ==> :

الى ماكرو الاول message وبالتالي تمت طباعتها عن طريق قناة الاخراج cout باستخدام الدليل text والذي يسبقه الرمز # لتحويل قيمة هـذا الدليـل الـى نـوع مـن السلسـة، تـلا ذلـك الرجـوع الـى الدالـة الرئيسـية ()main والمطالبـة بادخال قيمة المتغير الصحيح x ومن بعد ارسال الرسالة الثانية

The square of your value is ==>:

الـى مـاكرو message وطباعتهـا، ثـم الرجـوع ثانيـة الـى الدالـة الرئيســية وارسال قيمة المتغير x الى معامل ماكرو الاخير result وهو m حيث تمت طباعة مربع قيمته عن طريق cout، عموما عند التنفيذ وادخال القيمة العدديـة 5 للمتغير x سيعطي هذا البرنامج الناتج التالي:-

Enter your value ==> : 5

The square of your value is ==> : 25

مثال 7-18) البرنامج التـالي يوضـح اسـتخدام مـاكرو لتنفيـذ اكـثر مـن جملـة واحدة لاجل طباعة الاعداد من 1 إلى 4 مع مربعاتها.

```
#include<iostream.h>
#define loop for(int i=1;i<=n;i++)        \
{                                          \
```

```
	for(j=1;j<=i;j++)                  \
		cout<<k<<"  "<<k*k<<endl; \
		++k;                          \
}
main()
{
	int n=4, k=1;
	loop;
}
```

في الدالة الرئيسية يوجد أمران ، الاول الاعلان عن بعض المتغيرات ، والثاني هو استدعاء ماكرو تحت اسم loop ، والذي يحتوي على جملة for والتي بدورها تضم اكثر من سطر على ان يؤخذ في الاعتبار ان ينتهي كل سطر بالشرطة المائلة (\) كما هو مبين في هذا البرنامج.

11.7) الدوال الخطية Inline Functions

علاوة على الطريقة المعتادة لحل المسائل الصغيرة باستخدام الدوال العادية وايضا الماكرو اضافت لغة سي++ خاصية اخرى الى حل مثل هذه المسائل وهي الدالة الخطية التي يراعى فيها ان تضم عددا قليلا من الاوامر، وتتميز هذه الدالة بأنها لا تكلف المترجم الجهد الكبير في عملية الترجمة، فهي تستقبل الادلة من الدالة الرئيسية وتقوم بالتأكد من مطابقتها من حيث النوع والعدد.

الشكل العام :

```
inline ReturnType Identifier( ParameterDeclarationList)
{
	Statement(s)
}
```

يتم الاعلان عن الدالـة الخطيـة باستخدام الكلمـة المحجـوزة inline يتبعهـا نوع القيمة المرجعة والمعرف الـذي يمثـل اسـم هـذه الدالـة علـى ان توضـع الادلة مع اشهارها داخل القوسين ، يلي ذلك جملة أو عدد من الجمل التابعـة لهذه الدالة.

مثال 19-7) اعد كتابة البرنـامج بالمثـال (7-11) وظيفتـه اسـتدعاء دالـة خطيـة مهمتها حساب حجم الصندوق.

```
// Using an inline to calculate the volume of box
#include<iostream.h>
int inline box_volume(int length , int width , int height)
{
  return length*width*height ;
}
main()
{
      int L=8,W=5,H=2;
      int volume=box_volume(L,W,H);
      cout <<"The volume of a box with length "<<L<<endl
           <<"width "<<W<<" and height "<<H<<" is : "<<volume<<endl;
      return 0;
}
```

في البرنامج تم استدعاء الدالة الخطية box_volume عن طريق الجملة

```
int volume=box_volume(L,W,H);
```

التي بدورها مررت ثلاثـة مـن المتغيرات وهـي الطـول L والعـرض W والارتفاع H الى ما يقابلها من متغيرات داخل القوسين () بعد اسم الدالة التـي تبدأ بالكلمة inline وهي المتغيرات length و width و height علـى التوالـي ، تلا ذلك تنفيذ التعبير الحسابي للحصول على حجم الصندوق وبالتالي الرجوع بهذه القيمـة الـى الدالـة الرئيسية وتخصيصها للمتغير مـن النـوع الصحيـح volume وأخيراً يتم تنفيذ جملة الطباعة لاخراج جميع البيانات والمعلومات.

بعد اتمام تنفيذ هذا البرنامج سنحصل على النتائج التالية:-

```
The volume of a box with length 8
width 5 and height 2 is : 80
```

(12.7) تمارين Exercises

(1) اكتب دالة تحت اسم information () والتي تطبع اسمك ، عنوانك ، رقم هاتفك على أسطر مختلفة مع كتابة الدالة الرئيسية main() لاستدعاء الدالة المذكورة ثلاث مرات.

(2) اكتب برنامجا لايجاد اكبر واصغر قيمة باستخدام define لأربع قيم.

(3) اكتب برنامجا يقرأ قيمتين الاولي من النوع الصحيح الموجب n والثانية من النوع الحقيقي x ثم يستدعي دالة المعاودة الذاتية power التي مهمتها الرجوع بقيمة

$$\frac{1}{x^n}$$

(4) أكتب برنامجا يقرأ سلسلة حرفية ويستدعي دالة تختبر ما اذا كان الحرف المدخل من لوحة المفاتيح في الدالة الرئيسية موجودا في تلك السلسلة ثم يستدعي دالة أخرى مهمتها ايجاد عدد مرات ظهور هذا الحرف في نفس السلسلة ودالة ثالثة مهمتها ارجاع القيمة 1 اذا كان الحرف المدخل حرفاً ابجدياً كبير أو صغير وبالقيمة 0 في حالات الاخرى.

(5) أكتب برنامجا يستقبل متغيرين من نوع السلسلة str2,str1 ودالة تحت أسم Find تستقبل المتغيرين وترجع بموقع أول حرف من str1 في str2 أو ترجع بالقيمة 0 اذا كان str2 لا يحتوي str1 .

(6) أكتب دالة لها معاملان par2,par1 مهمتها نسخ كل محتويات par1 الى par2 فيما عدا الحروف A,F,N .

(7) المطلوب اعادة كتابة تمرين (6) باستخدام دالة المعادلة.

(8) اكتب برنامجا يستقبل قيمة من النوع الصحيح يمثل عدد الثواني ، ثم ارسال هذه القيمة الي دالة مهمتها تحويل هذه الثواني الى دقائق وساعات وايام والرجوع بهذه القيم الى نقطة الاستدعاء .

(9) تتبع البرامج التالية ثم حدد المخرجات .

(a)

```
#include<iostream.h>
main()
{
        double f(double a,int b);
        double a=5.5;
        int n=15;
        double result=f(a,n);
        cout<<"the result is "<<result<<endl;
        return 0;
}

double f(double c,int d)
{
        c=d+10;
        d=d%3;
        return c+d;
}
```

(b)

```
#include<iostream.h>
int a=20;
main()
{
        void fff();
        fff();
        fff();
        fff();
        cout<<"A="<<a<<endl;
}
void fff()
```

```
{
      int s=0;
      static int x=10;
      x++;
      s++;
      a++;
      cout<<"X="<<x<<"   S="<<s<<endl;
}
```

(10) المطلوب كتابة برنامج يقوم بقراءة سطر واحد، ثم يطبع هذا السطر في ترتيب عكسي باستخدام دالة المعاودة الذاتية PrintReverse.

(11) اكتب برنامجا لقراءة اطوال اضلاع مثلث c,b,a ثم احسب مساحته area عن طريق المعادلة التالية :-

$$area = \sqrt{s(s-a)(s-b)(s-c)}$$

حيث $s = (a+b+c)/2$

على ان تحسب قيمة s باستخدام ماكرو والمساحة باستخدام الدالة الخطية.

(12) ما هو ناتج تنفيذ البرنامج التالي اولا ثم اعد كتابته باستخدام الدالة الخطية ثانية.

```
#include<iostream.h>
#define mult(n1,n2)  n1*n2/n1+n2
main()
{
      int a=5,b=6;
      float c=2.5,d=10.0;
      int r1=mult(a,b);
      cout<<"A="<<a<<"  B="<<b<<" R1="<<r1<<endl;
      float r2=float (mult(c,d)/b);
      cout<<"A="<<a<<"  B="<<b<<" R1="<<r2<<endl;
}
```

(13) أكتب برنامجا لقراءة عدد صحيح number ثم كتابة دالة مهمتها ايجاد وطباعة الارقام الاولية الصماء prime اي التي لا تنقسم بغير باق الا على نفسها أو على واحد وذلك من 1 الي الرقم المدخل.

(14) صف مخرجات البرامج التالية :-

(a)

```
#include <iostream.h>
inline float in_line(const int a,const int b,const c)
{
	return (a%b+1*c+c%a);
}

main()
{
	int x=16,y=7,z=6;
	cout<<"The result of "<< in_line(x,z,y)<<endl;
}
```

(b)

```
#include <iostream.h>
int get_result(int,int);
main()
{
	int a=5,b=4;
	cout<<"The result is "<<get_result(a,b)<<endl;
	return 0;
}

int get_result(int c,int d)
{
	if(d>0)
	return get_result(c,d-1)+c;
}
```

(c)

```
#include<iostream.h>
main()
{
        int r=0;
        int cat(int x);
        for(int i=0;i<=4;i+=2)
        {
              r+=cat(i+1)+i;
              cout<<"R="<<r<<endl;
        }
}

int cat(int a)
{
        int rat(int y);
        int b=rat(a+1)+a;
        return (b);
}

int rat(int c)
{
        return c+c;
}
```

(d)

```
#include<iostream.h>
void one(void);
void two(void);
void three(void);
int value=10;
main()
{
        int value=50;
        {
              int value=25;
              cout<<"value="<<value<<<<endl;
        }
        cout <<"value="<<value<<" ===> variable inside main"<<endl;
```

```
    cout <<endl<<"***** value in first and second call *****"
        <<endl<<endl;
    for (int i=1;i<=2;i++)
    {
        one();
        two();
        three();
    }
    return 0;
}

void one(void)
{
    int value=77; // initialized each time one is called
    cout<<"value="<<value<<" ===> inside function one"<<endl;
    ++value;
}

void two(void)
{
    static int value=19; // static initializedation only
                         // first time two is called
    cout <<"value="<<value<<" ===> local static whene entering two"
        <<endl;
    ++value;
}

void three()
{
    cout <<"value="<<value<<" ===> global value whene entering three"
        <<endl<<endl;
}
```

(15) أكتب برنامجا باستخدام الدالة لحساب sum حيث

$$sum = \frac{2}{3} + \frac{3}{5} + \frac{4}{7} + \ldots + \frac{n+1}{2n+1}$$

الفصل الثامن: مؤثرات التعامل مع البت

تزودنا لغة سي++ بعدد من المؤثرات التي تتعامل مع اعداد النظام الثنائي أي البت (bit) والتي عن طريقها تتكون البايت (byte) وهي اساس تخزين أي قيمة حرفية في وحدة تخزين الحاسب ، وهو موضوع هذا الفصل .

1.8) الانظمة العددية Number Systems

قبل الدخول في مناقشة كيفية استخدام مؤثرات التعامل مع البت ، وحتى نسهل على القارىء الذي ليس له خلفية عن الانظمة العددية الاخرى والعلاقة فيما بينها والتي سوف يتناولها هذا الفصل ، وجب التطرق الى هذه الانظمة بشئ من الاختصار وهذه الانظمة هي:

1) النظام العشري Decimal System

يعتبر هذا النظام من اقدم الانظمة العددية التي عرفها الانسان ، وهو يستخدم في عد وحساب الاشياء ويتكون من الارقام التالية :-

9 , 8 , 7 , 6 , 5 , 4 , 3 , 2 , 1 , 0

وأساس (Base) هذا النظام 10 أي عشرة أرقام ومن هذه الارقام العشرة يمكن تكوين أي ترقيم في النظام العشري.

2) النظام الثنائي Binary System

ويعتبر من أنسب الانظمة العددية استعمالا وملاءمة للاستخدامات والتطبيقات في مجال الحاسبات، ولان معظم العناصر المستخدمة لتخزين البيانات في وحدة تخزين الحاسب هي ذات صفات ثنائية مزدوجة ، ولان جميع البوابات والدوائر المنطقية للحاسب تعمل بالكهرباء، لذا استخدم هذا النظام لتمثيل البيانات داخل ذاكرة الحاسب الذي اساسه 2 لانه يضم رقمين مختلفين هما الصفر 0 وهو يمثل حالة عدم وجود تيار كهربائي أي أن مفتاح الدائرة الكهربائية مغلق تماما، والواحد 1 وهو يمثل حالة وجود تيار كهربائي أي أن مفتاح الدائرة مفتوح تماما.

ومن هذا النظام جاءت كلمة بت (bit) وهي اختصار للعبارة (Binary digIT) والتي تعتبر أصغر معلومة يمكن تخزينها داخل الذاكرة وهي تمثل اما الرقم0 أو الرقم 1 ، وجاءت ايضا كلمة بايت (Byte) وهي التي تتكون من ثمانية ارقام أو جزئيات ثنائية (8 Bits) متجاورة من الآحاد والاصفار.

3) النظام الثماني Octal System

أساس هذا النظام الرقم 8 لانه يتكون من ثمانية ارقام وهي :-

7 , 6 , 5 , 4 , 3 , 2 , 1 , 0

4) النظام الستة عشرى Hexadecimal Sysytem

ويتكون من:

f,e,d,c,b,a,9,8,7,6,5,4,3,2,1,0

وسمي بهذا الاسم لانه يتكون من ستة عشر رمزاً وأساسه 16 وحتى تكون العلاقة واضحة بين هذه الانظمة العددية، فإن الجدول التالي يبين ذلك.

النظـــــــام			
السنة عشري	الثماني	الثنائي	العشري
0	0	0000	0
1	1	0001	1
2	2	0010	2
3	3	0011	3
4	4	0100	4
5	5	0101	5
6	6	0110	6
7	7	0111	7
8	10	1000	8
9	11	1001	9
a	12	1010	10
b	13	1011	11
c	14	1100	12
d	15	1101	13
e	16	1110	14
f	17	1111	15

ايضا لتسهيل توضيح بعض الارقـام فـي النظـام العشـري ومايكافؤهـا مـن أرقام في الانظمة العددية السابقة على أساس بايت واحدة فـإن الجـدول التـالي يبين هذه العلاقة.

النظـــــام			
العشري	الثنائي	الثماني	الستة عشري
128	1000 0000	200	80
5	0000 0101	5	5
15	0000 1111	17	f
93	0101 1101	135	5d
8	0000 1000	10	8
165	1010 0101	245	a5

ولمزيد الحصول على علاقة هذه الانظمة ببعضها البعض ، راجـع أسـكي (ASCII) بالملحق (2).

2.8) مؤثرات التعامل مع البت Bitwise Operators

وحتي يمكن معالجة وتخزين البيانات داخل وحدة تخزين الحاسب الآلـي، يتحتم علينا معرفة المؤثرات التي تتعامل مع هذا النوع من البيانـات وهـو مـا يعرف بمؤثرات التعامل مع البت (Bit) .

3.8) مؤثرات الازاحة أو النقل Shift Operators

هناك مؤثران يستخدمان للازاحة أو النقل تزودنا بها لغة سي++ وهما :-

المؤثر	معناه
>>	مؤثر الازاحه الي اليمين shift right
<<	مؤثر الازاحه الي اليسار shift left

يأخذ مؤثر الازاحة احدى الصيغتين الآتيتين :-

variable = variable OP expression ;

أو

variable OP = expression ;

حيث:

variable اسم المتغير.

OP مؤثر الازاحة الى اليسار أو الى اليمين.

expression التعبير الذي يحدد عدد الخانات التي ستزاح يسارا أو يمينا.

لغة سي++ لها من الميزات التي تمكنها من تحويل قيم حرفية الى ما يقابلها في الانظمة العددية العشري والثماني والستة عشري باستخدام المؤثرات hex,oct,int ، ولتحويل القيمة العددية الصحيحة الى ما يقابلها بالنظام الثنائي تتبع البرنامج التالي:-

مثال 8-1) البرنامج التالي يبين كيفية تحويل الاعداد في النظام العشري الى ما يقابلها في النظام الثنائي.

```
#include<iostream.h>
main()
{
        unsigned num,m=128;  // place 1 in leftmost postion
        for (int k=1;k<=3;k++)
        {
            cout<<"\nEnter a decimal number ==>:";
            cin>>num;
            cout<<num<<" in binary ==>";
            for(int i=1;i<=8;++i)
            {
                cout.put(m&num ? '1' : '0');
                if(i % 4==0)
```

```
                cout<<"     ";
            num<<=1;
        }
    }
    return 0;
}
```

تنفيذ البرنامج وادخال ثلاثة قيم من النوع الصحيح يولد الآتي :-

```
Enter a decimal number ==>:32
32 in binary ==> 0010   0000
Enter a decimal number ==>:21
21 in binary ==> 0001   0101
Enter a decimal number ==>:128
128 in binary ==> 1000   0000
```

تم الحصول على النظام الثنائي باستخدام جملة for الثانية التي احتوت على الجملتين :-

الاولى: ضمت دالة اخراج الحرفيات وفيها المؤثر الشرطي (?) وهي

```
cout.put(m&num ? '1' : '0');
```

حيث خصصت القيمة 128 للمتغير m الصحيح بدون اشارة وهذا يعني ان هذا المتغير يساوي بايت واحدة أي 8 ثنائيات ويحتوي في اقصى يساره على البت 1 وهو يكافيء 0000 1000 بالنظام الثنائي و 0x80 بالنظام الستة عشري وخصصت للتغير num القيمة 32 وهي تساوي 0000 0010 بالنظام الثنائي.

تتم عملية الشرط المنطقي m & num بين البت في اقصى اليسار لكل من المتغيرين m و num، ويكون الناتج اما صحيحا وبالتالي يطبع 1 أو خاطئا فيطبع 0.

الثانية: ضمت مؤثر الازاحة أي ازاحة خانة واحدة للمتغير num الى ناحية اليسار

```
num<<=1;
```

فتصبح قيمته 0000 0100 بعد الازاحة بدلا من 0000 0010 قبل الازاحة ، وهكذا حتى تتم عملية تكرار الحلقة ، وفيما يلي تتبع للحلقه for وللمتغيرات num,m,i والشرط m&num على النحو التالي :-

I	m	num	m&num
1	128	32	0
2	128	64	0
3	128	128	128
4	128	256	0
5	128	512	0
6	128	1024	0
7	128	2048	0
8	128	4096	0

في كل مرة تتكرر فيها جملة for نلاحظ الزيادة في قيمة المتغير num نتيجة ازاحته خانة واحدة الى اليسار.

مثال 8-2) البرنامج التالي يبين كيفية ازاحة الحرف A الذي تعادل قيمته 0100 0001 بالنظام الثنائي عدد خانتين لليسار.

```
#include<iostream.h>
void bitfunction(unsigned);
main()
{
        char letter='A',char_out;
        cout<<endl<<"letter in binary ==> ";
        bitfunction(letter);
        cout<<" = "<<int(letter)<<" in decimal";
        char_out=letter<<2;
        cout<<endl<<"letter=letter<<2 ==> ";
        bitfunction(char_out);
        cout<<" = "<<int(char_out)<<" in decimal";
}
```

```
void bitfunction(unsigned letter)
unsigned m=128;
{
    for(int i=1;i<=8;++i)
    {
        cout.put(m&letter ? '1' : '0');
        if(i % 4==0)
            cout<<"  ";
        letter<<=1;
    }
}
```

letter in binary ==>0100 0001 = 65 in decimal

letter=letter<<2 ==>0000 0100 = 4 in decimal

هنا قيمة الحرف A قبل الازاحة بالنظام الثنائي هي 0001 0100 حسب جدول أسكي وقيمته بالنظام العشري تكافيء 65، بينما قيمته بعد الازاحة هي 0100 0000 أي القيمة 4 بالنظام العشري.

مثال 8-3) يمكن ازاحة نفس الحرف A عدد خانتين الى اليمين بتغيير جملة الازاحة في البرنامج السابق لتصبح

```
char_out=letter>>2;
```

قيمة الحرف A قبل الازاحة هي 0001 0100، وعليه سوف تصبح قيمته الازاحة 0000 0001 أي القيمة 16 بالنظام العشري.

مثال 8-4) اذا كانت لدينا الاشهارات للمتغيرات التالية :-

```
int A,B;
char C;
```

واسندت لكل من هذه المتغيرات القيم التالية :-

```
A=10; B=93;
C='Z';
```

فالجدول التالي يبين بعض عمليات الازاحة لهذه المتغيرات :-

النظام				
الثنائي	الستة عشري	الثماني	العشري	التعابير
0000 1010	a	12	10	A
0010 1000	28	50	40	A<<=2
0000 0101	5	5	5	A>>=1
0101 1101	5d	135	93	B
0000 0101	5	5	5	B>>=4
1010 0000	a0	240	160	B<<=5
0101 1010	5a	132	90	C
0000 0010	2	2	2	C>>=5
0100 0000	40	100	64	C<<=5

4.8) المؤثرات المنطقية Logical Operators

توجد أربعة مؤثرات منطقية مألوفة في لغة سي++ تتعامل مع البت وهي:-

المؤثر	معناه
&	مؤثر الضرب " و " and الناتج 1 اذا كان كل من العنصرين 1
|	مؤثر الجمع " أو " or الناتج 1 اذا كان احد العنصرين أو كليهما 1
^	مؤثر الجمع المنفرد " أو المقتصره " exclusive or الناتج 1 اذا كان احد العنصرين 1
~	مؤثر المتمم للواحد one's complement يعكس الاصفار الى آحاد والآحاد الى أصفار.

مثال 8-5) الجدول التالي يبين كيفية عمل مؤثرات البت بالنسبة للمتغيرين y,x

x	y	x&y	x^y	x|y	~x
1	1	1	0	1	-2
1	0	0	1	1	-2
0	1	0	1	1	-1
0	0	0	0	0	-1

مع مؤثر المتتم للواحد في حالة ان المتغير x له القيمة 1 في النظام الثنائي وهي 0001 0000 يتم تحويل كل الاصفار لتصبح آحادا وكل الآحاد لتصبح أصفارا وبالتالي نتحصل على القيمة 1110 1111 وهي تساوي 2- ، وبالمثل متمم القيمة 0 والتي تكافي 0000 0000 تصبح 1111 1111 وهي تساوي 1- .

نلاحظة عند استخدام جميع المؤثرات التي تتعامل مع البت أنها ذات عنصرين ماعدا مؤثر المتتم للواحد فهو ذو عنصر واحد ، وأولويات تنفيذها من أعلى الى أسفل حسب الاسبقية الموضحة بالجدول التالي :-

التنفيذ	المؤثر
من اليسار الى اليمين	~
من اليسار الى اليمين	>>
من اليسار الى اليمين	<<
من اليسار الى اليمين	&
من اليسار الى اليمين	^
من اليسار الى اليمين	|

مثال 8-6) على فرض انه خصصت للمتغيرين b,a قيمتان من النوع الصحيح هما 165,93 على التوالي ، المطلوب اجراء بعض العمليات باستخدام مؤثرات البت السابقة على هذين المتغيرين.

قبل التطرق الى استخدام هذه المؤثرات، نحن نعلم مما سبق ان هذه المؤثرات تستخدم على الاعداد ذات النظام الثنائي (Binary System) الذي يعمل به جهاز الحاسب الآلي، وبالتالي فان الرقمين 93 و 165 يمثلان بالنظام الثنائي على اساس بايت واحدة بالثنائيات 1101 0101 و 0101 1010 وبالنظام الستة عشرى بالارقام 5d و a5 على التوالي انظر ملحق (2).

1) استخدام مؤثر الضرب (&)

بالنظام الثنائي	بالنظام الستة عشري
a = 0101 1101	a = 5d
b = 1010 0101	b = a5
-----------------------	--------------
a&b = 0000 0101	a&b = 05

2) استخدام مؤثر الجمع (|)

بالنظام الثنائي	بالنظام الستة عشري
a = 0101 1101	a = 5d
b = 1010 0101	b = a5
---------------------	------------
a | b = 1111 1101	a | b = fd

3) استخدام مؤثر الجمع المنفرد (^)

بالنظام الثنائي

```
a    = 0101 1101
b    = 1010 0101
-------------------
a^b  = 1111 1000
```

بالنظام الستة عشري

```
a    = 5d
b    = a5
----------
a^b  = f8
```

4) يستخدم مؤثر المتمم للواحد (~) على عنصر واحد فقط

```
a    = 0101 1101
---------------------
~a   = 1010 0010
```

مثال 8-7) البرنامج المتكامل التالي يوضح كيفية استخدام المؤثـرات المنطقيـة مع مؤثر الازاحة على نفس قيم المتغيرين b,a السابقين مع التنفيذ والشرح.

```
#include<iostream.h>
void bitfunction(unsigned );
main()
{
	unsigned ,a=93,b=165,c;
	cout <<"\t\tLOGICAL OPERATORS"<<endl
		<<"\t\t****************************" <<endl
		<<"\tExpression   Decimal  "
		<<"  Hexadecimal    Binary \n" <<endl
		<<"\t ---------------------------------------------" <<endl;
	for(int j=1;j<=6;j++)
	{
		switch(j)
		{
			case 1:  c=a;
				cout <<"\t A\t\t"<<dec<< a
					<<"\t  "<<hex<< a<<"\t\t";
				bitfunction(c ); break;
```

```
                case 2:  c=b;
                         cout <<endl<<"\t B\t\t "<< dec<< b
                              <<"\t  "<<hex<< b<<"\t\t";
                         bitfunction(c ); break;
                case 3:  c=~b;
                         cout <<endl<<"\t ~B\t\t"<< dec<< ~b
                              <<"\t  "<<hex<< ~b<<"\t";
                         bitfunction(c ); break;
                case 4:  c=a & b;
                         cout <<endl<<"\t A&B\t\t"<< dec<< c
                              <<"\t  "<<hex<< c<<"\t\t";
                         bitfunction(c ); break;
                case 5:  c=a|b;
                         cout <<endl<<"\t A|B\t\t"<< dec<< c
                              <<"\t  "<<hex<< c<<"\t\t";
                         bitfunction(c ); break;
                case 6:  c=a^b;
                         cout <<endl<<"\t A^B\t\t"<< dec<< c
                              <<"\t  "<<hex<< c<<"\t\t";
                         bitfunction(c ); break;
            }
        }
        getch();
        return 0;
}

void bitfunction(unsigned number)
{
        unsigned m=128;
        for(int i=1;i<=8;++i)
        {
            cout.put(m&number ? '1' : '0');
            if(i % 4==0)
                   cout<<"  ";
            number<<=1;
        }
}
```

وهذه هي النتائج عند تنفيذ هذا البرنامج

LOGICAL OPERATORS

****

Expression	Decimal	Hexadecimal	Binary	
A	93	5d	0101	1101
B	165	a5	1010	0101
~B	65370	ff5a	0101	1010
A & B	5	5	0000	0101
A l B	253	fd	1111	1101
A ^ B	248	f8	1111	1000

استعملت في هذا البرنامج متغيرات من النوع الصحيح بدون اشارة والحصول على الجدول الاعلى الذي ضم التعبير وقيمته بالنظام العشري والستة عشرى والثنائي من خلال جملة switch اعتمادا على قيمة العدادز بالحلقة for وفي كل حالة تم ايجاد قيمة المتغير c وطباعته بالنظام الستة عشرى باستخدام الاداة hex والنظام العشري باستخدام الاداة int وتم الحصول على قيمة c بالنظام الثنائي باستخدام الدالة bitfunction() بحيث مررت إلى دليلها المتغير number قيمة المتغير c من الدالة الرئيسية وهي 93 والتي تساوي 0101 1101 بالنظام الثنائي.

بعد الطباعة تمت ازاحة خانة واحدة للمتغير number الى ناحية اليسار وعليه اصبحت قيمته بعد الازاحة 1010 1011، وهكذا حتى تتم عملية التكرار الحلقة for والرجوع الى نقطة الاستدعاء بالدالة الرئيسية.

مثال 8-8) نختم هذه الامثلة ببرنامج كامل مهمته توضيح كيفية استخدام المؤثرات الخاصة بالازاحة.

```
#include<iostream.h>
void bitfunction(unsigned);
main()
{
	unsigned n=5;
	char char_in='A',char_out;
	cout <<" Expression   Decimal  "
		<<"    Action      Binary"<<endl
		<<" ---------------------------------------"<<endl;
	for(int k=1;k<=2;k++)
	{
		for(int j=1;j<=3;j++)
		{
			switch(j)
			{
				case 1: char_out=char_in;
					cout <<endl<< char_in<<"\t\t"
						<<int(char_in)<<"\t"<<" unshifted"<<"\t";
					bitfunction(char_out); break;
				case 2: char_out=char_in<<n;
					cout <<endl<<char_in<<"<<="<< n<<"\t\t"
						<<int(char_out)<<"\t left "<< n<<"\t\t";
					bitfunction(char_out); break;
				case 3: char_out=char_in>>n;
					cout <<endl<<char_in<<">>="<< n<<"\t\t"
						<<int(char_out)<<"\t right  "<< n<< "\t";
					bitfunction(char_out); break;
			}
			n-=2;
		}
		char_in='Z';
		n=7;
	}
	return 0;
}

void bitfunction(unsigned letter)
{
	unsigned m=128;
	for(int i=1;i<=8;++i)
	{
```

```
            cout.put(m&letter ? '1' : '0');
            if(i % 4==0)
                  cout<<"   ";
            letter<<=1;
      }
}
```

في بداية هذا البرنامج تم اسناد الحرف A للمتغير char_in من النوع الحرفي جاءت بعدها جملة الطباعة لاخراج عنوان الجدول ، تلا ذلك استخدام جملتي التكرار for:

الاولي: مهمتها اجراء مؤثرات الازاحة على المتغير char_in عندما يكون المتغير به القيمة الحرفية A

الثانية: ووظيفتها اجراء بعض عمليات الازاحة على المتغير char_in عن طريق جملة switch اعتمادا على قيمة المتغير j كالآتي:-

اذا كانت قيمته تساوي 1 عندها اطبع قيمة المتغير char_in بدون ازاحة مع انتقال التحكم إلى الجملة التي تلي القوس المغلق لجملة switch وهي طرح العدد 2 من قيمة المتغير n لتصبح 3 بدلا من 5 واذا كانت قيمة j تساوي 2 عندها يتم ازاحة قيمة المتغير char_in عدد n من الخانات ناحية اليسار، وأخيراً ازاحة نفس قيمة المتغير char_in خانة واحدة ناحية اليمين وفي كل حالة تطبع النتائج مع استدعاء الدالة bitfunction() والتي مهمتها استقبال قيمة المتغير char_in واسنادها الى دليلها letter وبالتالي طباعة هذه القيمة بالنظام الثنائي.

وقبل الرجوع الى جملة التكرار الاولى يتم تغيير قيمة المتغير char_in بالقيمة الحرفية Z بدلا من القيمة الحرفية A مع اسناد القيمة 7 للمتغير n بقصد اجراء نفس المؤثرات السابقة على هذه القيمة.

والجدول التالي هو ناتج تنفيذ هذا البرنامج وقت تنفيذه :-

expression	Decimal	Action	Binary
A	65	unshifted	0100 0001
A<<=3	8	left 3	0000 1000
A>>=1	32	right 1	0010 0000
Z	90	unshifted	0101 1010
Z<<=5	64	left 5	0100 0000
Z>>=3	11	right 3	0000 1011

5.8) تمارين Exercises

(1) اذا كان المتغيران الصحيحان y,x يحملان القيمتين 7,4 على التوالي ، أي 0000 0111,0000 0100 بالنظام الثنائي ، المطلوب اجراء المؤثرات الخاصة بالبت على هذين المتغيرين ، مع تحويل الناتج الى ما يقابله بالنظامين الثماني والستة عشرى .

(2) على فرض ان

```
int x=25,y=20;
char a='A',b='F';
```

اوجد قيمة التعابير التالية:-

(a) x l y	(b) a & b
(c) x ^ b	(d) y>>2
(e) x<<2	(f) ~(y l b)

(3) اكتب برنامجا مهمته كتابة الاعداد العشرية الصحيحة 2 ,4 ,8 ,16 ,32 ,64 بنظام العد الثنائي والثماني والستة عشري.

(4) اكتب برنامجا يعكس كل بت في البايت التالية ، فمثلا

10011101 تصبح 10111001

(5) المطلوب كتابة برنامج لازاحة قيمة الحرف m في نظام الشفرة (ascii) وهي القيمة العددية 01101101 في النظام الثنائي أربعة مواقع الى اليسار وثلاثة مواقع الى اليمين مع تحويل الناتج الى ما يقابله في بقية الانظمة في كل مرة.

(6) اذا اعطيت جملة الاعلان يتبعها الدالة كالآتي:

```
int a=14, b=7;
```

```
(void bitfunction(int number )
{
        int m=128;
        for(int i=1;i<=8;++i)
        {
             cout.put(m&number ? '1' : '0');
             if(i % 4==0)
             cout<<"  ";
             number<<=1;
        }
}
```

المطلوب ايجاد ما يطبع في كل حالة من الحالات التالية :-

(a)
```
cout<<endl<<"A="<<dec<<a<<"  B="<<hex<<a<<"  ";
bitfunction(a);
```

(b)
```
cout<<endl<<"A="<<dec<<~a<<"  B="<<hex<<~a<<"  ";
bitfunction(~a);
```

(c)
```
int c=a & b;
cout<<endl<<"A="<<dec<<c<<"  B="<<hex<<c<<"  ";
bitfunction(c);
```

(d)
```
int c=a | b;
cout<<endl<<"A="<<dec<<c<<"  B="<<hex<<c<<"  ";
bitfunction(c);
```

(e)
```
int c =a ^ b;
cout<<endl<<"A="<<dec<<c<<"  B="<<hex<<c<<"  ";
bitfunction(c);
```

(f)
```
int c =a>>2;
cout<<endl<<"A="<<dec<<c<<"  B="<<hex<<c<<"  ";
bitfunction(c);
```

(g)
```
int c =b<<3;
cout<<endl<<"A="<<dec<<c<<"  B="<<hex<<c<<"  ";
bitfunction(c);
```

الفصل التاسع : المصفوفات

عند تنفيذ البرامج التي تعالج بيانات كثيرة بواسطة الحاسب فاننا مضطرون الى استخدام عدد كبير من المتغيرات لكي نتمكن من تخزين هذه البيانات الأمر الذي يجعل البرنامج طويلا ومملا ومعقدا في بعض الأحيان والسبب الرئيسي في ذلك هو أننا لا نستطيع حفظ سوي قيمة واحدة فقط في المتغير الواحد.

ولحل هذه المعضلة يجب اللجوء الى طريقة أخرى ألا وهي استخدام المصفوفات (Arrays) للتعامل مع البيانات ، وهذه المصفوفات عبارة عن تركيبة بيانية مندرجة تحت اسم واحد يستطيع المبرمج استخدامها كمتغيرات وذلك بتقسيم كل متغير الى عدد من العناصر المتسلسلة المحتوية على قيم مختلفة ومتشابهة من حيث النوع.

1.9) المصفوفات ذات البعد الواحد One-dimensional Arrays

المصفوفة هي عبارة عن صف واحد أو عمود واحد يحتوي على مجموعة من عناصر البيانات المتحدة النوع والاسم وشكلها العام هو :

```
type array-name [size];
```

حيث:

type يمثل نوع المصفوفة، array-name يمثل اسمها، size يمثل عدد عناصرها، وقد يكون رقما أو متغيرا أو ثابتا له قيمة صحيحة موجبه.

مثال 9-1) اشهار

```
int list[6];
```

يعني الاعلان عن مصفوفة أحادية تحت اسم list تحتوي على 6 عناصر من النوع الصحيح int أول هذه العناصر يبدأ من list[0] وآخرها list[5] ، وقد تكون بعناصر هذه المصفوفة القيم العددية الموضحة كالآتي :-

11	-8	77	10	0	-1
list[0]	list[1]	list[2]	list[3]	list[4]	list[5]

حيث يشار الى أي عنصر من هذه العناصر باستخدام اسمها أولا ودليل العنصر داخل المصفوفة الموجود بين القوسين [] ثانيا ، وعليه فان :

list[2] بها القيمة 77 ، في حين list[5] يوجد بها القيمة 1- .

مثال 9-2) المطلوب شرح البرنامج التالي :-

```
#include<iostream.h>
main()
{
	int list[6];
	for(int i=0;i<=5;i++)
		list[i]= i;
	for(i=0;i<=5;i++)
		cout<< list[i]<<"  ";
	return 0;
}
```

في هذا البرنامج تم الاعلان عن مصفوفة صحيحة ذات بعد واحد تحت اسم list وشحنها بالقيم الصحيحة الموجبة من 0 الى 5 عن طريق الدليل المتغير i مع جملة for ، وبالتالي احتوت المصفوفة على 6 قيم صحيحة تمت طباعتها كالآتي :-

```
0 1 2 3 4 5
```

يجب الأخذ في الاعتبار بداية ونهاية الدليل المستخدم مع المصفوفة ، على ألا يتعدى الدليل (حجمها - 1) والا كان الناتج خاطئا.

مثال 9-3) لنفرض أننا نريد قراءة رقم الطالب ودرجته المتحصل عليها في الامتحان وذلك لفصل به 5 طلبة ، والمطلوب طباعة رقم الطالب ودرجته مع الفرق بين كل درجة والمتوسط .

```
// this program inputs a sequence of test scores,
// finds their average and displays the scores together
// with their defferences from the average.
#include<iostream.h>
main()
{
        const int number_of_student=5;
        float score[number_of_student];
        long i,id_no[number_of_student];
        float sum=0.0;
        cout <<"Type id# and score of "<<number_of_student
              <<" students please : "<<endl;
        for(i=0;i<number_of_student;i++)  // loop to input id# and scores
        {
              cout <<"Enter id# and score "<<i+1<<" ==>: ";
              cin>>id_no[i]>>score[i];
              sum+=score[i];
        }
        avg=sum/number_of_student;
        cout <<"\n------------------------------------------------" <<endl;
              // display id#, scores, and differences from average
        cout.setf(ios::showpoint | ios::fixed);
        cout.precision(1);
        for(i=0;i<number_of_student;i++)
              cout <<endl<<"ID# "<<id_no[i] <<" score = "
                    <<score[i] <<" different ==>: "<<score[i]-avg;
        cout<<endl<<"The average =====> "<<avg;
        return 0;
}
```

اذا ما نفذ هذا البرنامج سينتج عنه اخراج الرسالة التالية :-

Type id# and score of 5 students please :

طالباً ادخال رقم قيد الطالب ودرجته لعدد 5 طلبة ، وقد تكون المدخلات والمخرجات كما يلي :-

```
Enter id# and score 1 ==>:983001 76
Enter id# and score 2 ==>:973223 46
Enter id# and score 3 ==>:963106 92
Enter id# and score 4 ==>:963076 52
Enter id# and score 5 ==>:983024 63
---------------------------------------------------
ID# 983001 score = 76.0 different ==>: 10.2
ID# 973223 score = 46. 0 different ==>: -19.8
ID# 963106 score = 92 different ==>: 26.2
ID# 963076 score = 52.0 different ==>: -13.8
ID# 983024 score = 63.0 different ==.: -2.8
The average ====>: 65.8
```

بدأ البرنامج بالاعلان عن نوع واسم المصفوفة وحجمها وهي

```
float score[number_of_student]
```

يعني اختير الاسم score للدلالة على انها مصفوفة من النوع الحقيقي لاستعمالها كمخزن لحفظ درجة الطالب ، على ان يكون عدد الدرجات التي تخزن في هذه المصفوفة لا يتجاوز 5 درجات .

ويمكن اعادة كتابة الاعلان السابق كالآتي :-

```
float score [5];
```

ولكن الطريقة الاولى هي الأحسن ، حيث يمكن تغيير عدد الطلبة بالفصل بتغيير قيمة الثابت number_of_student الى أي قيمة يريدها المبرمج مع ترك بقية جمل البرنامج كما هي، كما اعلن عن مصفوفة اخرى من النوع الصحيح الطويل تحت اسم id_no لاستعمالها لحفظ ارقام قيد الطلبة وبالتالي يمكن الرجوع الى قيم هذه المصفوفات وقت الحاجة اليها.

بعد الاعلان عن المتغيرات المناسبة جاءت بعض الجمل التوضيحية تلاها استخدام جملة for لغرض ادخال البيانات الخاصة بكل طالب ومن ثم تخزين هذه البيانات في الأماكن التي يشير اليها الدليل i في المصفوفة المعنية بداية من 0 الى 4 مع حساب مجموع درجات الطلبه الخمسة ، بعد حساب المتوسط ، استعملت جملة for لاخراج البيانات التي تم ادخالها سالفا مصحوبة بالفارق بين درجة كل طالب ومتوسط هؤلاء الطلبة بالفصل.

مثال 9-4) ترتيب عناصر أي مصفوفة تصاعديا أو تنازليا يعتبر شائع الاستعمال في الحاسب الآلي مثل ترتيب أرقام قيد الطلبة أو ترتيب أسعار بضاعة معينة ، البرنامج التالي يقوم بقراءة عدد من القيم الحقيقية وتخزينها في مصفوفة ومن ثم ترتيبها ترتيبا تصاعديا.

```
#include<iostream.h>
#include<stdlib.h>
main()
{
      float temp,a[11];
      int n,i,j,k,n1,n2,counter=1;
      cout<<"Type number of elements : ";
      cin>>n;
      if(n>10 || n<=0)
            exit(0);
      for(i=1;i<=n;i++)
      {
            cout<<"A["<<i<<"] = ";
```

```
        cin>>a[i];
    }
    n1=n-1;
    cout<<endl<<"Steps of sorting the array as following :";
    for(i=1;i<=n1;i++)
    {
        n2=i+1;
        for(j=n2;j<=n;j++)
        if(a[j]-a[i]<0)
        {
            temp=a[i];
            a[i]=a[j];
            a[j]=temp;
            cout<<endl<<"\t step"<<counter<<"-->";
            for(k=1;k<=n;k++)
                cout<<a[k]<<" ";
            counter++;
        }
    }
    cout <<endl<<endl
        <<"The sorted array as following :";
    for(i=1;i<=n;i++)
        cout<<endl<<"A["<<i<<"] = "<<a[i];
    return(0);
}
```

عند اظهار الرسالة المناسبة والمطالبة بادخال عدد القيم المراد ترتيبها وليكن 6 قيم يتم بعدها ادخال هذه القيم على النحو التالي :-

```
Type number of elements: 6
A[1]= 3.9
A[2]= 8.4
A[3]= 6.6
A[4]= 4.8
A[5]= 7.1
A[6]= 5.2
```

يلي ذلك الرسالة التي تدل على خطوات تبديل هذه القيم داخل المصفوفة وهي كالتالي :-

Steps of sorting the array as following :

STEP 1 ==> 3.9 6.6 8.4 4.8 7.1 5.2
STEP 2 ==> 3.9 4.8 8.4 6.6 7.1 5.2
STEP 3 ==> 3.9 4.8 6.6 8.4 7.1 5.2
STEP 4 ==> 3.9 4.8 5.2 8.4 7.1 6.6
STEP 5 ==> 3.9 4.8 5.2 7.1 8.4 6.6
STEP 6 ==> 3.9 4.8 5.2 6.6 8.4 7.1
STEP 7 ==> 3.9 4.8 5.2 6.6 7.1 8.4

أخيرا يأتي نتائج ترتيب قيم عناصر المصفوفة تصاعديا كما يلي :-

The sorted array as following :
A[1]= 3.9
A[2]= 4.8
A[3]= 5.2
A[4]= 6.6
A[5]= 7.1
A[6]= 8.4

هنا تم الاعلان عن مصفوفة ذات بعد واحد من نفس نوع البيانات المدخلة، واسمها a بحيث يمكنها استيعاب عدد 10 قيم حقيقية.

بعد عملية ادخال البيانات تم استخدام جملة for المتداخلة والتي مهمتها استبدال قيمتي a[i] و a[j] كلما كانت قيمة العنصر الأول أكبر من قيمة العنصر الثاني ، ومن هنا يبدأ البرنامج بالبحث عن أصغر قيمة في المصفوفة ،

ولتحديد ذلك تقارن قيمة a[1] وهي القيمة 3.9 مع قيمة a[2] وهي 8.4 وبما أنهما مرتبتان تصاعديا فلا يتم التبادل بينهما ، وتبقي القيم في مكانها ، أما في الحالة الثانية ، فتتم مقارنة قيمة a[2] وهي 8.4 مع قيمة a[3] وهي 6.6 وبما انهما غير مرتبتين فيتم التبادل بينهما في الأماكن عن طريق متغير ثالث مؤقت temp لاستخدامه في عملية التحويل في أماكن المصفوفة كالآتي :-

```
temp=8.4
a[2]=6.6
a[3]=8.4
```

وبالتالي فان قيمة المصفوفة الموجودة في العنصر الثاني تصبح 6.6 على حين تصبح قيمة المصفوفة الموجودة في العنصر الثالث 8.4، وهكذا تتكرر عملية التبديل كلما دعت الضرورة لذلك حتي نهاية جملة for المتداخلة.

2.9) المصفوفات ذات البعدين Tow-dimensional Arrays

المصفوفة ذات البعدين تتكون من مجموعة الصفوف (rows) والأعمدة أي Columns وقد تحتوي على بيانات من النوع الصحيح int أو الحقيقي float أو غيرها، والصيغة العامة لهذه المصفوفة هي:

```
type array-name[size1][size2];
```

حيث:

type نوع المصفوفة ، array-name اسمها.

size1 الدليل الأول وهو يشير الى عدد صفوف المصفوفة.

size2 الدليل الثانى وهو يشير الى عدد الأعمدة بالمصفوفة.

مثال 9-5) خذ مثلا هذا الاعلان

int mat[3][4];

فهو يعني ان المصفوفة mat هي من النوع الصحيح وتحتوي على ثلاثة صفوف وأربعة أعمدة أي أن بها 12 عنصرا (3X4) أول هذه العناصر هو mat[0][0] وآخرها هو mat[2][3] .

مثال 9-6) اكتب برنامجاً كاملاً مهمته قراءة عدد من القيم الصحيحة مع تخزينها في مصفوفة ذات بعدين ثم طباعة هذه المصفوفة مع مجموع قيم عناصر كل صف من صفوف المصفوفة.

```
#include<iostream.h>
main()
{
	int array[3][4];
	cout<<"Enter the elemnt of the array : "<<endl;
	for(int i=0;i<3;i++)
		for(int j=0;j<4;j++)
		{
			cout<<"ARRAY ["<<i<<","<<j<<"] ==>: ";
			cin>>array[i][j];
		}
	cout <<endl<<"The array looks like :"<<endl
		<<"========================="<<endl<<endl;
	for(i=0;i<3;i++)
	{
		int sum=0;
		for(j=0;j<4;j++)
		{
			cout<<"\t"<<array[i][j];
			sum+=array[i][j];
		}
		cout<<"\tSum of row "<<i+1<<" ==> "<<sum<<endl<<endl;
	}
	return(0);
}
```

بعد تنفيذ هذا البرنامج وإظهار السطر

Enter the elemnt of the array :

والذي يطالب بادخال القيم الىعناصر المصفوفة وقد يكون مشابها للآتي:-

ARRAY [0,0]= -2
ARRAY [0,1]= 1
ARRAY [0,2]= 6
ARRAY [0,3]= 7
ARRAY [1,0]= -5
ARRAY[1,1]= 2
ARRAY [1,2]= 5
ARRAY [1,3]= 8
ARRAY [2,0]= -3
ARRAY [2,1]= 3
ARRAY [2,2]= 4
ARRAY [2,3]= 9

The array looks like :

==================================

-2	1	6	7	sum of row 1 ===> 12
-5	2	5	8	sum of row 2 ===> 10
-3	3	4	9	sum of row 3 ===> 13

في هذا البرنامج تم الاعلان عن مصفوفة array ذات بعدين من النوع الصحيح مع تحديد عدد صفوفها وأعمدتها.

استخدم في هذا البرنامج جملة for المتداخلة الاولى تمثل عدد الصفوف والثانية تمثل عدد الاعمدة وذلك لقراءة القيم وتخزينها في المصفوفة array، ونقول هنا ان القيم تم تخزينها في المصفوفة باتجاه الصف (row-wise) أي

الصف الاول ثم الصف الثاني فالثالث وهكذا، ايضا استخدمت جملة for الموالية حيث كان مهمتها:

1) تنفيذ الامر

```
int sum=0;
```

أي اشهار المتغير sum مع اعطائه القيمة المبدئية 0 .

2) تنفيذ جملة for الداخلية التي مهمتها

- طباعة العنصر الاول من المصفوفة array باتجاه الصف بالامر

```
cout<<"\t"<<array[i][j];
```

- حفظ مجموع عناصر هذا الصف في المتغير sum بالامر

```
sum+=array[i][j];
```

3) طباعة قيم المتغير sum مع الانتقال الى سطر جديد لطباعة الصف الموالي باستخدام الجملة

```
cout<<"\tSum of row "<<i+1<<" ==> "<<sum<<endl<<endl;
```

3.9) المصفوفات الحرفية Character Arrays

المصفوفة الحرفية هي التي تكون من النوع الحرفي string ومنها :-

1) المصفوفة ذات البعد واحد One-dimensional Array

يجدر بنا قبل التطرق الى استخدام المصفوفة من هذا النوع ، ان نذكر القارىء الكريم إنه سبق لنا مناقشة المتغير الحرفي ، فالاعلان

```
char a;
```

يعني أن المتغير الحرفي a قابل ان يخزن فيه حرف واحد فقط في كل مرة يتم استعماله.

كما سبق وان تطرقنا الى الاعلان عن متغير بالشكل

char sample[10];

فهذا الوصف يعني ان المتغير sample هو اسم لمصفوفة من النوع الحرفي وقد حدد لها 10 أماكن في الذاكرة ، وان المترجم يضيف في نهاية هذه المصفوفة رمز النهاية ('\0') وعليه فعند الاعلان عن مصفوفة من هذا النوع يجب أن يؤخذ في الاعتبار ترك مكان لهذا الرمز .

مثال 9-7) اشرنا فيما سبق الى أن المترجم يضيف الرمز ('\0') في نهاية المصفوفة من النوع الحرفي ، البرنامج التالي يستفيد من ذلك ليقرأ أي سلسلة حرفية لا يتعدي طولها 50 حرفا ثم يقوم باحصاء عدد حروفها.

```
#include<iostream.h>
main()
{
	const int size=50;
	char str[size];
	int n,counter;
	cout<<"Enter your string ==>:";
	cin.get(str,size);
	counter=0;
	for(n=0;str[n]!='\0';n++)
	counter++;
	cout <<endl<<"Your string has the length ==>: "<<counter
		<<" Characters";
		return 0;
}
```

البرنامج اذا ما نفذ وادخلت السلسلة:

Enter your string ==>: This is only a test

سوف يعطي السطر:

Your string has the length 19 Characters

تم ادخال السلسلة الحرفية عن طريق دالة العضو cin.get وتخزينها في المتغير str من نوع المصفوفة ، تلا ذلك تكرار زيادة القيمة 1 الى المتغير counter عن طريق جملة for في كل مرة لم يتحقق فيها شرطها str[n] !='\0' ، وتتوقف هذه الاعادة عند وصول التحكم الى الرمز ('\0') الذي يدل على نهاية حروف السلسلة بالمصفوفة ، عندها يتم طباعة طولها الذي يمثله المتغير counter.

2) المصفوفة ذات البعدين Two-dimensional Array

يمكن انشاء مصفوفة ذات بعدين لتخزين الحرفيات فيها .

فالاعلان التالي :-

```
char list_name[50][20];
```

يحجز الحاسب داخل ذاكرته مصفوفة تحت اسم list_name بحيث تتكون من 50 سلسلة حرفية كل منها لا يزيد حجمها على 20 حرفا ، و يمكن الوصول الى أي عنصر من عناصر هذه المصفوفة بتحديد الدليل الاول فقط، فمثلا للوصول الى الاسم الثامن بالمصفوفة ، يكتب الامر:

```
cout<<list_name[7];
```

مثال 9-8) ادخال مجموعة من الاسماء وتخزينها في مصفوفة حرفية ذات بعدين ، ثم طباعة هذه المصفوفة ، هي مهمة البرنامج الموالي :-

```
#include<iostream.h>
main()
{
	const NUMBER=5;
	const LENGTH=20;
	char name[NUMBER][LENGTH];
```

```
    cout<<"Please enter names ==>:"<<endl;
    for(int m=0;m<NUMBER;m++)
        cin.getline(name[m],LENGTH,'\n');
    cout<<endl<<"Hi The names you typed as following :"<<endl;
    for(m=0;m<NUMBER;m++)
        cout<<name[m]<<endl;
    return 0;
}
```

بدأ البرنامج بالاعلان عن المصفوفة الحرفية

name[NUMBER][LENGTH]

من ذات البعدين مهمتها استقبال وتخزين عدد 5 اسماء كل واحد من هذه الأسماء ينبغي ألا يتعدي 20 حرفا، ولادخال البيانات تم استخدام المتغير m كعداد والناتج من عملية جملة for كدليل للمصفوفة name والذى يمثل ترتيب الأسماء، وبنفس الطريقة تمت طباعة هذه الاسماء، حيث استخدم الدليل الايسر الذي يمثل ترتيب الاسماء بالمصفوفة فقط، فاذا ما ادخلت الاسماء التالية عند تنفيذ البرنامج

```
Please enter names ==>:
ANAS MOHAMED
RAEF BASHIR
KADIJA AHMED
FADI SAAD
KALED RAMADAN
MAHA A. JALAL
```

عندها سيطبع البرنامج الآتي :-

```
Hi The names you typed as following :
ANAS MOHAMED
RAEF BASHIR
```

KADIJA AHMED

FADI SAAD

KALED RAMADAN

MAHA A. JALAL

مثال 9-9) اكتب برنامجاً كاملاً يقوم بقراءة اسم الطالب وعدد المقررات التي درسها بالقسم خلال الفصل مع رقم ودرجة كل مقرر من هذه المقررات ثم طباعة تقرير يضم كل البيانات المدخلة مع طباعة رقم أعلى وأدني درجة تحصل عليها الطالب ومتوسطه العام وذلك لعدد لا يتجاوز 5 مقررات.

```
#include<iostream.h>
#include<conio.h>
main()
{
        char name[20];
        char code[5][5];
        int number_of_course;
        float score[5],sum=0.0;
        clrscr();
        cout<<"Please enter names ==>:";
        cin.getline(name,20,'\n');
        cout<<"Enter number of courses ==>: ";
        cin>>number_of_course;
        for(int j=0;j<number_of_course;j++)
        {
             cout<<"----------------------------"<<endl;
             cout<<"Enter score ["<<j+1<<"] ==>: ";
             cin>>score[j];
             sum+=score[j];
             cout<<"Enter code course ["<<j+1<<"] ==>: ";
             cin>>code[j];
        }
        cout <<endl<<"The student name is [ "<<name;
        cout <<" ] has the following informations : "
             <<endl<<"------------------------------"<<endl;
        for(j=0;j<number_of_course;j++)
        {
```

```
        cout<<endl<<"Course code ["<<j+1<<"] is "<<code[j];
        cout<<"\t The score is "<<score[j];
    }
    float max_grd=score[0],min_grd=score[0];
    float max=0,min=0;
    for(j=1;j<number_of_course;j++)
    {
        if(score[j]<min_grd)
        {
            min_grd=score[j];
            min=j;
        }
        if(score[j]>max_grd)
        {
            max_grd=score[j];
            max=j;
        }
    }
    cout <<endl<<endl<<"maximum score = "<<max_grd
        <<"\t code = "<<code[max]
        <<endl<<"minamum score = "<<min_grd
        <<"\t code = "<<code[min]<<endl;
    float avg=sum/number_of_course;
    cout<<"\t\t\t average is = "<<avg;
    getch();
}
```

تم الاعلان عن المصفوفة ذات بعد واحد name من النوع الحرفي ، تستخدم لتخزين اسم الطالب الذي يجب ألا يزيد عدد حروفه على 20 حرفا والمصفوفة code ذات بعدين من النوع الحرفي على أساس تخزين رقم المادة بطول 5 حروف ولعدد 5 مقررات والمصفوفة score من النوع الحقيقي لتخزين درجات الطالب.

بعد ادخال اسم الطالب وعدد المقررات المسجل فيها ، جرى استخدام الدليل الاول j من المصفوفة الحرفية code ليشير الى تخزين رقم المادة في ذلك المكان ومن ثم ايجاد أعلى وأدني درجة.

واخيرا اذا ما نفذ هذا البرنامج وجري ادخال اسم الطالب

Please enter names ==>: MAHMOUD BASHIR

يتبعه ادخال عدد المقررات وليكن 3 كالآتي :-

Enter number of courses ==>: 3

وبالتالي ادخال الدرجة ورقم المقرر كما يلي:-

Enter score [1] ==>: 80

Enter code course [1] ==>: CS100

Enter score [2] ==>: 70

Enter code course [2] ==>: MA101

Enter score [3] ==>: 90

Enter code course [3] ==>: CS111

عندها يتم اخراج كل المعلومات التي تخص الطالب بالصورة التالية:-

The student name is [MAHMOUD BASHIR] has the following informations:

Course code [1] is CS100 The score is 80.00

Course code [2] is MA101 The score is 70.00

Course code [3] is CS111 The score is 90.00

maximum score = 90.00 code = CS111

minamum score = 70.00 code = MA101

average is = 80.00

4.9) القيم الأبتدائية Initial Values

يمكن تخصيص قيم ابتدائية لأي مصفوفة عند الاعلان عنها على أن تكون هذه القيم من نفس نوع المصفوفة ، واذا لم يتم تخصيص قيم مبدئية لعناصر المصفوفة ، عندها تأخذ المصفوفة قيما عشوائية أو أصفارا ، وقد تأخذ قيم المصفوفة الشكل التالي:

```
type array_name [size]={ list-of-constant-values };
```

حيث يلي اسم وحجم المصفوفة array_name القوسان {} بينهما القيم المبدئية للمصفوفة.

مثال 9-10) خذ مثلا الاعلان التالي :-

```
int X[5]={3,-5,6,2,7};
```

فهو يكافىء الاعلان والجمل التخصيصية الآتية :-

```
int X[5];
X[0]=3;
X[1]=-5;
X[2]=6;
X[3]=2;
X[4]=7;
```

وكما هو واضح فان الاعلان الاول هو الانسب في حالة معرفة قيم المصفوفة مسبقا.

مثال 9-11) البرنامج التالي مهمته شحن مصفوفة احادية بعدد من القيم الصحيحة باستخدام الثابت وطباعتها.

```
#include<iostream.h>
main()
{
        const int SIZE=3;
        const mat[SIZE]={10,20,30};
        for(int i=0;i<SIZE;i++)
              cout<<mat[i]<<' ';
        return 0;
}
```

نلاحظ في هذا البرنامج ان القيم التي تم تخصيصها للمصفوفة mat هي قيم صحيحة وبالتالي فان المصفوفة ينبغي ان تكون صحيحة ايضا، البرنامج سيطبع الآتي عند التنفيذ :-

10 20 30

هنا اذا غيرنا قيمة الثابت SIZE لتصبح 5 ، حينها سيطبع البرنامج القيم الثلاث السابقة المخصصة للمصفوفة مع طباعة بقية القيم اصفارا.

مثال 9-12) البرنامج الموالي يبين تخصيص قيم من النوع الحرفي لمصفوفة من نفس النوع.

```
#include<iostream.h>
main()
{
        const int SIZE=11;
        char A[SIZE]={'I',' ','l','i','k','e',' ','C','+','+','\0'};
        for (int i=0;i<SIZE;i++)
              cout<<A[i];
        return 0;
}
```

هنا تم الاعلان عن مصفوفة أحادية طولها 11 وهي قابلة لأن تحتوي على قيم حرفية ، وفي نفس الوقت تم تخزين الحروف المبدئية التي تقع بين قوسي

الفئة {} أي الجملة I like C++ في المصفوفة الحرفية A والتي عددها 11 حرفا من بينها فراغان والرمز ('0\') ليدل على نهاية هذه الحروف بداية من الحرف الاول I ثم الثاني وهو فراغ والثالث الحرف l وهكذا ثم طباعة محتوي المصفوفة عن طريق جملة for.

في حالة تخصيص قيمة SIZE أقل من المطلوب ينتج عنها خطأ والسبب ضرورة أن يكون حجم المصفوفة مساويا لعدد الحروف المطلوب تخزينها مع اضافة مكان للرمز ('0\') .

قد يأخذ الاعلان عن المصفوفة A في المثال السابق الشكل التالي :-

```
char A []={'I',' ','L','I','K','E',' ','C','+', '+', '\0'};
```

هنا لم نحدد طول المصفوفة ، لأن المترجم سيقوم بحساب هذا الطول ، مما يريح المبرمج من هذه المهمة.

مثال 9-13) في البرنامج التالي يتم تخصيص أيام الأسبوع المعروفة في مصفوفة وطباعتها.

```
#include<iostream.h>
main()
{
	const int length=4;
	const int day_of_week=7;
	char week_days[day_of_week][ length]=
		{"SAT","SUN","MON","TUE","WED","THU","FRI"};
	cout<<"The week days are :"<<endl;
	for(int i=0;i<day_of_week;i++)
		cout<<week_days[i]<<",   ";
	return 0;
}
```

تم الاعلان عن مصفوفة من النوع الحرفي ذات بعدين تحت اسم week_days ، البعد الأول به الدليل day_of_week الذي يمثل عدد أيام الأسبوع، وقد يلغى هذا الدليل كما سبق وأن ذكرنا مع المصفوفة ذات البعدين وذلك على النحو التالي :-

week_days[][length]

أما البعد الثاني الذي به الدليل length فهو لتحديد عدد حروف اليوم الواحد، أي ثلاثة حروف اضافة الى رمز النهاية ('\0') ، اذا ما نفذ هذا البرنامج باي من الطريقتين السابقتين سيعطي النتائج المشابهة للآتي :-

The week days are :

SAT, SUN, MON, TUE, WED, THU, FRI

5.9) تمرير المصفوفات الى الدوال Passing Arrays to Functions

ذكرنا سابقا أنه عند معالجة بيانات كثيرة جدا يتحتم علينا استخدام المصفوفات ليسهل معها عملية تخزين هذه البيانات في متغيرات قليلة ، ولكي يصبح البرنامج سهل الكتابة والمتابعة والفهم يجب أن يقسم البرنامج الى عدد من الدوال الفرعية ، وبالتالي ينبغي أن نأخذ في الاعتبار كيفية تمرير قيم عناصر المصفوفة الى أي دالة اخرى ، وقد يحدث أن تتم عملية تمرير البيانات الى الدوال باستخدام المتغيرات الخارجية (global variables).

مثال 9-14) المطلوب كتابة برنامج كامل يتم فيه قراءة قيم صحيحة وتخزينها في مصفوفة ذات بعد واحد ثم يقوم باستدعاء دالة فرعية مهمتها تخزين هذه القيم بالعكس في مصفوفة اخرى والرجوع بها وطباعتها في الدالة الرئيسية.

```
#include <iostream.h>
const int M=3;
int i,k,A[M],B[M];
main()
{
      void invers_array();
      for( i=0;i<M;i++)
      {
            cout<<"Enter A["<<i<<"] ==>: ";
            cin>>A[i];
      }
      invers_array();
      cout<<endl<<"Array after reversed is :";
      for( i=0;i<M;i++)
            cout<<endl<<"B["<<i<<"] ==>: "<<B[i];
      cout<<endl;
      getch();
      return 0;
}
void  invers_array()
{
      k=M-1;
      for( i=0;i<M;i++)
            B[i]=A[k--];
}
```

عند تنفيذ هذا البرنامج وادخال ثلاث قيم من النوع الصحيح كما يلي :-

ENTER A[0]: 5

ENTER A[1]: 6

ENTER A[2]: 7

حينها سيكون الناتج هو الآتي :-

Array after reversed is :

B[0] = 7

B[1] = 6

B[2] = 5

لاحظ هنا أن الاعلان عن المصفوفتين والمتغيرين B,A والمتغيرين k,i من النوع الصحيح ، قبل بداية الدالة الرئيسية يجعلها متغيرات خارجية ويمكن التعامل معها في أي دالة تكون تابعة لهذه الدالة الرئيسية.

عندما استدعيت الدالة invers_array() وهي دالة بدون معاملات، اي انها لم تقم بتمرير قيم عناصر المصفوفة من الدالة الرئيسية الى هذه الدالة، بل تم هذا العمل عن طريق المصفوفة الخارجية A والتي هي معروفة في هذه الدالة، بحيث وضعت قيم عناصرها معكوسة في المصفوفة الخارجية B والتي هي أيضا معروفة في الدالة الرئيسية عن طريق جملة for حيث تم الرجوع بها الى الدالة الرئيسية وطباعتها هناك.

قد يحدث تمرير البيانات من الدالة الرئيسية الى أي دالة فرعية أخري بالاعلان عن المصفوفة في كلا الطرفين بحيث تكون مطابقة من حيث النوع والطول.

مثال 9-15) خذ مثلا الدالة الرئيسية التالية التي مهمتها القيام باستدعاء ثلاث دوال مختلفة.

```
#include <iostream.h>
#define MAX 10
main()
{
      float mat[MAX];
      read_mat(MAX,mat);
      sort_mat(MAX,mat);
      prnt_mat(MAX,mat);
}
```

تأخذ الدالة الفرعية الاولي read_mat الشكل التالي :-

```
read_mat(k,cat)
float cat[k];
int k;
{
      . . .
}
```

في بداية هذه الدالة تم الاعلان عن مصفوفة cat وهي تعني أنها مصفوفة حقيقية وطولها k من العناصر أي لها نفس نوع وطول المصفوفة المقابلة لها عند نقطة الاستدعاء ، وقد تكون مهمة هذه الدالة هي قراءة k من القيم الحقيقية وتخزينها في المصفوفة cat .

اما الدالة الفرعية الثانية sort_mat فقد تكون بالشكل الآتي :-

```
sort_mat(k,rat)
float rat[];
int k;
{
      . . .
}
```

الاعلان هنا عن المصفوفة اختلف عن مثيلتها في الدالة السابقة حيث الاعلان rat[] يبين أنها مصفوفة من النوع الحقيقي ولكن غير محددة الطول، حيث يخزن فيها كل العناصر التي مررت اليها من المصفوفةmat بالدالة الرئيسية ، وهي طريقة مقبولة وقد يكون مهمة هذه الدالة ترتيب عناصر هذه المصفوفة ، أما الدالة prnt_mat فقد يكون مهمتها اخراج عناصر المصفوفة.

مثال 9-16) يمكن اعادة كتابة البرنامج المكتوب بالمثال (9-4) وذلك لاعادة ترتيب عدد من القيم ترتيبا تصاعديا.

```
#include <stdlib.h>
#include <iostream.h>
int i,j;
```

```
read_mat (int k,float cat[]);
prnt_mat(int size, float fat[]);
sort_mat(int m,float rat[]);
main()
{
        int n;float temp,mat[10];
        cout<<"Type number of elements: ";
        cin>>n;
        if(n>10||n<=0)
              exit(0);
        read_mat(n,mat);
        sort_mat(n,mat);
        prnt_mat(n,mat);
        return 0;
}

              // This function for input data data
read_mat (int k,float cat[])
{
        for(i=0;i<k;i++)
        {
              cout<<"mat["<<i<<"]=";
              cin>>cat[i];
        }
        return 0;
}

              // This function for sorting data
sort_mat(int m,float rat[])
{
        float temp;int n1,n2,n3,counter=1;
        n1=m-1;
        for(i=0;i<n1;i++)
        {
              n2=i+1;
              for(j=n2;j<m;j++)
              {
                    if(rat[j]-rat[i]<0)
                    {
                          temp=rat[i];
                          rat[i]=rat[j];
```

```
                rat[j]=temp;
                cout<<endl<<" STEP "<<counter <<"-->";
                for(n3=0;n3<m;n3++)
                      cout<<" "<<rat[n3];
                counter=counter+1;
            }
        }
    }
    return 0;
}

            // This function for output data
prnt_mat(int size,float fat[])
{
    cout<<endl<<"The sorted array as following:"<<endl;
    for(i=0;i<size;i++)
    cout<<endl<<"mat["<<i<<"]="<<fat[i];
}
```

قبل بداية الدالة الرئيسية تم الاعلان عن المتغيرين j,i على أساس انهما متغيران خارجيان مع العينة (Prototype) للدوال الثلاثة المراد استخدامها في هذه الدالة الرئيسية .

وفي بداية الدالة الرئيسية اعلن عن المصفوفة ذات البعد الواحد mat من النوع الحقيقي مع ادخال عدد عناصرها واستدعاء الدوال التالية:-

الدالة الأولى: read_mat (int k,float cat[]) ومهمتها استقبال k من القيم الحقيقية عن طريق لوحة المفاتيح وتخزينها في المصفوفة تحت اسم cat غير المحددة الطول.

الدالة الثانية: sort_mat(int m,float rat[]) وظيفتها استقبال جميع قيم عناصر المصفوفة mat التي تم الحصول عليها من خلال الدالة read_mat وتخزينها في المصفوفة rat ، وترتيب m من القيم

ترتيبا تصاعديا مع طباعة خطوات الترتيب ، وتمرير قيم المصفوفة rat الى المصفوفة mat عند نهاية الدالة.

الدالة الثالثة: prnt_mat(int size,float fat[]) وظيفة هذه الدالة هي استقبال قيم العناصر المرتبة في المصفوفة mat من الدالة الرئيسية الى المصفوفة المقابلة fat، وطباعة كل القيم بالمصفوفة التي حجمها size من العناصر.

عند تنفيذ البرنامج وادخال البيانات المناسبة ، سيعطي نفس النتائج بالبرنامج المشار اليه في هذا المثال .

مثال 9-17) البرنامج التالي يقوم بعملية ادخال مصفوفتين مربعتين من النوع الصحيح ثم اجراء عمليات الضرب على هاتين المصفوفتين.

في هذا البرنامج دالتان :-

1) الدالة الأولى input_mat لها دليلان y,x وكلاهما مصفوفة ذات بعدين من النوع الصحيح قابلة لاستقبال عدد sizecol من الاعمدة وعدد غير محدود من الصفوف وبعد ادخال عدد الصفوف وعدد الاعمدة للمصفوفتين تمت قراءة وتخزين قيم عناصر كل مصفوفة على حدة باستخدام جملة for المتداخله واخيرا تمرير قيم المصفوفتين y,x الى المصفوفتين المقابلة لهما عند نقطة الاستدعاء بالدالة الرئيسية.

الدالة الثانية mult_ mat مهمتها ضرب المصفوفتين حيث تم تحديد نوع وطول الصفوف والاعمدة لكل مصفوفة، تلا ذلك استخدام جملة if أي مقارنة طول عدد العناصر بالصف بالنسبة للمصفوفة الاولى مع عدد العناصر بالعمود بالمصفوفة الثانية ، وعند عدم التساوي تظهر الرسالة الدالة على ذلك

يليها الخروج من هذه الدالة والرجوع الى الدالة الرئيسية ، اما في حالة عدم التساوي فحينها تخزن المصفوفة الخارجية c بالأصفار قبل أن تستخدم في عملية تخزين حاصل ضرب المصفوفتين باستخدم جملة for أكثر من مرة للحصول على الناتج ، يلي ذلك طباعة هذه المصفوفة والرجوع الى نقطة الاستدعاء بالدالة الرئيسية، وفيما يلي البرنامج كاملاً بعد ضم الدالتين مع الدالة الرئيسية في برنامج واحد بالترتيب التالي :-

```
#include <iostream.h>
const int sizecol=10,sizerow=10;
int i,j,row,col;
int c[sizerow][sizecol];

input_mat(int x[][sizecol], int y[][sizecol])
{
	cout<<"How many rows ? ==>: ";
	cin>>row;
	cout<<"\nHow many columns ? ==>: ";
	cin>>col;
	if ((row>sizerow) || (col>sizecol))
	exit (0);
	cout<<"Enter elements of array one:\n";
	for(i=0;i<row;i++)
	     for(j=0;j<col;j++)
	          cin>>x[i][j];
	cout<<"Enter elements of array two:\n";
	for(i=0;i<row;i++)
	     for(j=0;j<col;j++)
	          cin>>y[i][j];
	return 0;
}

mult_mat(int x[sizerow][sizecol], int y[sizerow][sizecol])
{
	int k;
	if(row!=col)
	{
	     cout<<"Sorry error data\n";
```

```
            cout<<"Row and column are not equal\n";
            exit(0);
        }
        for(i=0;i<row;i++)
        for(j=0;j<col;j++)
        {
            c[i][j]=0;
            for(k=0;k<row;k++)
            c[i][j]=c[i][j]+x[i][k]*y[k][j];
        }
        cout<<"Multipliction of two arrays:\n\n";
        for(i=0;i<row;i++)
        {
            for(j=0;j<col;j++)
                cout<<c[i][j]<<"\t";
            cout<<"\n\n";
        }
}

main()
{
        int a[sizerow][sizecol],b[sizerow][sizecol];
        input_mat(a,b);
        mult_mat(a,b);
        return 0;
}
```

نلاحظ هنا ان الدوال الفرعية وضعت قبل الدالة الرئيسية وهذا جائز كما ذكرنا سابقا ، وعليه لم تستخدم عينات لهذه الدوال في هذا البرنامج، وفي بداية الدالة الرئيسية تم الاعلان عن المصفوفتين b,a على أساس انهما مصفوفتان معروفتان في هذه الدالة فقط، واذا ما نفذنا هذا البرنامج وتم الرد على الرسالتين بادخال عدد الصفوف وعدد الأعمدة للمصفوفتين على النحو التالي :-

```
How many rows ? ==> : 3
How many columns ? ==> 3
```

يتبعها المطالبة بادخال قيم عناصر المصفوفة الأولي

Enter elements of array one :

6 6 6

5 5 5

4 4 4

وقيم عناصر المصفوفة الثانية

Enter elements of array two :

2 2 2

3 3 3

1 1 1

بعدها يرد الحاسب بالنتائج المشابهة للآتي:

Multiplication of two arrays :

36 36 36

30 30 30

24 24 24

(6.9) تمارين Exercises

(1) المطلوب تعريف مصفوفة من نوع صحيح int بها القيم المبدئية الآتية:-

(20,80,40,10,33,70,55)

(2) المطلوب قراءة مصفوفة ذات بعدين مع ايجاد مجموع قيم العناصر أعلى القطر الرئيسي من المصفوفة ومجموع قيم الغناصر أسفل القطر الرئيسي من المصفوفة.

(3) المطلوب كتابة برنامج كامل لقراءة مصفوفة ذات بعدين mat بها n × n عنصرا ثم استخدام دالة لايجاد حاصل ضرب عناصر القطر الرئيسي أي

mat(1,1)*mat(2,2)*...mat(n,n)

(4) اكتب برنامجا يبحث عن قيمة X من النوع الصحيح في مصفوفة LIST ذات بعد واحد طولها 100 ، فاذا كانت القيمة X موجودة عندها يطبع مكان وجودها ، واذا لم تكن موجودة تطبع الرسالة التالية:-

The value of X not found

(5) ما هو ناتج تنفيذ البرامج التالية على جهاز الحاسب ؟

(a)

```
#include <iostream.h>
int get_result(int [],int);
main()
{
        int cat[8]={4,8,-3,4,1,3,-7,2};
        cout<<"The result is "<<get_result(cat,8)<<endl;
        return 0;
}

int get_result(int rat[],int length)
{
        if(length==1)
             return rat[0];
```

```
      else
            return get_result(rat,length-1)+rat[length-1];
}
```

(b)

```
#include<iostream.h>
main()
{
      int sum1=0,sum2=0;
      int arr[3][4]={
                        {11,2,3,14},
                        {15,6,7,18},
                        {9,3,10,12}
                     };
      for(int i=0;i<3;i++)
            for(int j=0;j<4;j++)
                  if(arr[i][j] %3 !=0)
                        sum1+=arr[i][j];
                  else
                        sum2+=arr[i][j];
      cout<<"sum1="<<sum1<<"  sum2="<<sum2<<endl;
      cout<<arr[i][j]<<endl;
}
```

(6) اكتب برنامجا رئيسيا مستدعيا فيه دالة بالشكل التالي :-

int max(int size , int mat[])

مهمتها ارجاع اكبرعنصر في المصفوفة mat والتي طولها size من العناصر.

(7) في البرنامج بالتمرين (5) فقرة (b) ، ما هي القيمة المخزنة بالموقع arr[2][3] مع ذكر موقع القيمة 6 في المصفوفة arr

(8) المطلوب كتابة برنامج لطباعة المصفوفة ذات بعدين وذلك باستخدام القيم المبدئية ، مرة وجمل التكرار مرة اخرى ، على أن تكون المصفوفة عند طباعتها بالشكل التالي :-

```
1 0 0 0 0
1 1 0 0 0
1 1 1 0 0
1 1 1 1 0
1 1 1 1 1
```

(9) أعطيت مصفوفة ذات بعدين matr بها 5 X 5 عنصرا المطلوب استخدام دالة واحدة مهمتها الرجوع بالآتي :-

* مجموع عناصر هذه المصفوفة.
* أكبر عنصر في هذه المصفوفة.
* متوسط العناصر الموجبة.

(10) اكتب برنامجاً كاملاً يتم فيه تعريف المصفوفات بالشكل التالي :-

```
int two_array[10] [5] , one_array[10];
```

يتبعها ادخال البيانات الخاصة بالمصفوفة two_array ثم يقوم البرنامج باستدعاء دالة تحت اسم sum_col مهمتها ايجاد مجموع كل عمود من اعمدة المصفوفة two_array وتخزينها بالمصفوفة one_array والرجوع بها وطباعتها بالبرنامج الرئيسي.

(11) نفس البيانات المعطاة في تمرين (9) مع كتابة دالتين الاولي مهمتها تخزين جميع عناصر المصفوفة السالبة في مصفوفة أحادية NEGTV بينما الثانية مهمتها تخزين جميع عناصر المصفوفة الموجبة في مصفوفة أحادية POSTV.

(12) أكتب برنامجا لقراءة مصفوفة عددية ذات بعد واحد ثم رتب هذه الاعداد ترتيبا تصاعديا، وأيضاً أوجد وسط (Median) هذه المصفوفة أي العدد الاوسط في حالة أن المصفوفة تحتوي علىعدد فردي من الاعداد أو متوسط العددين الاوسطين في حالة أنها تحتوي على عدد زوجي من العناصر.

(13) أعد حل التمرين السابق لايجاد منتصف مصفوفة بها عدد زوجي من العناصر وليكن 16 .

(14) المصفوفتان xx و yy كل واحدة منهما تحتوي على N عنصرا في ترتيب تصاعدى ، المطلوب دمج المصفوفتين في مصفوفة واحدة zz على ان يكون عناصرها في ترتيب تصاعدى ، فمثلا اذا كانت xx=1,3,9 و yy=3,6 عليه تكون zz=1,3,3,6,9

(15) اكتب برنامجا يستقبل مصفوفة صحيحة حجمها 10 مع استدعاء دالة مهمتها استقبال هذه المصفوفة والرجوع بقيم عناصر هذه المصفوفة غير المتكررة في ترتيب تصاعدي ، فمثلا اذا كانت هذه القيم كالتالي :-

5 12 9 26 5 77 60 18 20 9

عندها تكون العناصر كالآتي :-

5 9 12 18 20 26 60 77

الفصل العاشر: المؤشـــرات

يقصد بالمؤشر في علم الحاسب الآلي الشي الذي يشـير الـى موقـع متغير وليس لاسم متغير ولها دور مهم في تمرير القيم من الدالة الرئيسية الى الدالة الفرعية وبالعكس بالاضافة إلى أغراض أخرى نتناولها في هذا الفصل.

1.10) الاعلان عن المؤشر Pointer Declaration

في البرامج السابقه تم اشهار المتغيرات من النوع الحرفي مثل

```
char a;
```

وهو يعني ان المتغير a من النوع الحرفي، ويمكن تخزين حرف واحد فقط في كل مرة بموقع هذا المتغير.

والآن حان الوقت للتعمق في موضوع اشهار متغيرات من نوع المؤشر الحرفي ، الذي يمكننا من الاشارة الى بداية السلسلة الحرفية string في ذاكرة الحاسب وشكله العام هو

```
Type_Name *Variable_Name1,*Variable_Name2,...;
```

خذ مثلا الاعلان

```
char *ch;
```

هنا الرمز (*) يشير الى موقع بداية القيمة المخزنة في العنوان الذي يشـير اليه المؤشر ch ، في حين يعتبر المتغير ch مؤشرا pointer يشير الـى موقـع بداية قيمة من النوع الحرفي char .

مثال 10-1) ما الذي نلاحظه في البرنامج الموالي.

```
#include<iostream.h>
main()
{
	char *ch="Nice to meet you ";
	cout<<ch<<endl;
	cout<<*ch;
}
```

نلاحظ هنا الآتي :-

1) تم الاعلان عن المتغير ch والذي يعتبر مؤشرا إلى قيمة من النوع الحرفي char بداية علامة (*).
2) تم تخصيص عنوان بداية السلسلة وهو الحرف N للمتغير ch .
3) اضافة الرمز ('\0') في نهاية السلسلة ليشير الى نهايتها من قبل النظام.
4) جملة الطباعة الاولي سينتج عنها طباعة السلسلة حرفا حرفا ،بداية من الحرف N وحتي الوصول الى رمز النهايه ('\0') .
5) جملة الطباعة الثانية انتجت طباعة الحرف الاول N والذي يشير اليه المؤشر ch فقط.

وعليه سيكون الناتج عند التنفيذ مشابها للآتي :-

```
Nice to meet you
N
```

مثال 10-2) البرنامج التالي يوضح لنا طباعة السلسلة حتي نهايتها التي تتمثل في الرمز ('\0')

```
#include<iostream.h>
main()
{
	char *STR="Why don't you go near the water ?";
	while(*STR !='\0')
	cout<<*STR++;	//  increment STR to point to the next character
}
```

بعد الاشارة الى بداية السلسلة

Why don't you go near the water ?

الذي يمثله الحرف الاول منها W جاءت جملة while التي مهمتها طباعة الحرف الاول من السلسة مع زيادة القيمة 1 للمتغير STR في كل مرة وبالتالي الانتقال الى الحرف الموالى بهذه السلسلة ، وهكذا تتكرر نفس الخطوات السابقة حتي يصبح الشرط ('0\'=! STR *) صحيحا أي انه تم الوصول الى نهاية السلسلة، عندها يقف البرنامج والتنفيذ يعطي الناتج التالي :-

Why don't you go near the water ?

ما هو ناتج تنفيذ البرنامج السابق اذا ما استبدلت جملة التخصيص لتصبح:

*STR = "Why don't you go \0near the water ?"

الجواب هو الناتج التالي :-

Why don't you go

أي طباعة السلسلة حرفا بعد حرف حتي الوصول الى الرمز('0\') والموجود قبل كلمة near.

يمكن ان يؤشر المؤشر إلى قيمة من النوع الصحيح ، فالاعلان التالي:-

```
int *ptr;
```

يعني ان ptr مؤشر يشير الى موقع به قيمة من النوع الصحيح ، في حين أن الجملة

```
int *ptr=55;
```

تعني انه قد تم تخزين القيمة الصحيحة 55 في الموقع المشار اليه بالمؤشر ptr من النوع الصحيح ، في حين أن الجملة

```
ptr=&k;
```

تعني ان ptr مرجع لـ k أي (Refering to k) أو يحتوي على عنوان k أي (Containing address of k) أو مؤشر لـ k أي (Pointing to k).

خذ مثلا اذا كان لدينا الاعلان

```
double x,y,*ptr;
```

فالجملتان

```
ptr=&x;
y=*ptr;
```

مشابهتان للجملة

```
y=*&x;
```

مثال 10-3) ما هو ناتج تنفيذ البرنامج التالي ؟.

```
#include <iostream.h>
main()
{
      int c=77;
      int *a;           // pointer to an integer
      int x=55;
      a=&x;      // assign address of x to a
      cout<<"x==>:"<<x<<"  *a==>:"<<*a<<endl;
      a=&c;      // assign address of c to a
      cout<<"X==>:"<<x<<"  *a==>:"<<*a<<"  c==>:"<<c<<endl;
}
```

في هذا البرنامج تم اشهار المتغير a مؤشرا يؤشر الى قيمة من النوع الصحيح ، وخصصت القيمة 55 الى المتغير الصحيح x في حين ان الجملة

a=&x;

تعني تخصيص موقع أو عنوان المتغير x الى المتغير a أو بمعني آخر يشير المؤشر a ليدل على موقع الذاكرة المخصص للمتغير x ، وعليه ستكون نتيجة تنفيذ جملة الطباعة كما يلي :-

x==>:55   *a==>:55

وبنفس الكيفية في الجملة

a=&c;

نجد ان a يؤشر الى c وتكون قيمة كل واحدة منهما هي القيمة 77 في حين بقيت x بنفس قيمتها السابقة كما يوضحه الناتج بالسطر التالي :-

x==>:55   *a==>:77   c==>:77

مثال 4-10) نوع آخر من استخدام المؤشرات يوضحه البرنامج التالي :-

```
#include <iostream.h>
#include <string.h>
main()
{
	char ch='A',*p,str[30];
	p=&ch;
	cout <<"*p ==>"<<*p<<"  *p+1 ==>"<<char(*p+1)
		<<" *p+2 ==>"<<char(*p+2)<<endl;
	str[0]='X',str[1]='Y',str[2]='Z',str[3]='\0';
	p=str;
	cout <<"str ==>"<<str<<" While *p ==>"<<*p<<"  *p+1 ==>"
		<<char(*p+1)<<" *p+2 ==>"<<char(*p+2)<<endl;
	strcpy(str,"Boy go home now please");
	cout<<str<<endl;
	p+=4;
	for( ;*p !='\0' ; ++p)
	{
		if(*p == 'o')
```

```
            *p='*';
        if(*p == ' ')
            *p='\n';
    }
    cout<<str;
}
```

ينتج عن هذا البرنامج وقت التنفيذ الآتي :-

```
*p ==>A  *p+1 ==>B  *p+2 ==>C
str ==>XYZ  While *p ==>X  *p+1 ==>Y  *p+2 ==>Z
Boy go home now please
Boy g*
h*me
n*w
please
```

في هذا البرنامج اعلن عن المتغير ch مـن النـوع الحرفـي وخصصـت لـه القيمة 'A' بينما المتغير p هو مؤشر إلى قيمة من النوع الحرفي ، ايضا اعلـن عـن المتغير str مـن نـوع السلسلة الحرفيـة مـع حفـظ القيـم 'X','Y','Z' فـي الأماكن الثلاثة الأولى والرمز '\0' في المكـان الرابـع مـن هـذا المتغيـر ، تـلا ذلك الامر p=&ch حيـث &ch هـو عنـوان الذاكـرة للمتغيـر ch ، وعنـوان ch يوضـع في p ، فـي حيـن ان p* يحتـوي علـى مـا يؤشـر اليـه p ، وبمـا ان p يؤشر الى ch و ch يحتـوي علـى الحـرف 'A' ، عليـه سـتطبع قيمـة p* وهـو الحرف A ، يلي ذلك طباعة الحرف B لان التعبير p+1* هو اكبر بواحد مـن p* أيضا التعبير p+2* هو اكبر باثنين من p* عليه سيطبع الحرف C .

الامر   p=str ;

هنا يحتوي المؤشر str على المؤشر المتغير p ، وعليه تمت طباعة محتوي str وهي القيمة XYZ ، يلي ذلك طباعة قيم التعابير *p+2,*p+1,*p وهي الاحرف Z,Y,X على التوالي.

في البرنامج ايضا تم نسخ السلسلة الحرفية Boy go home now please الى المتغير str وطباعتها واضافة العدد 4 الى قيمة المؤشر p وبذلك يؤشر الى المكان الذي يبعد 4 حروف من البداية في الذاكرة ، وبما أن p يؤشر الى str[0] ، عليه فان جملة for ستبدأ بمعالجة السلسلة بداية من الحرف g حيث يتم تغيير الحرف o بالرمز (*) إذا كانت قيمة p تؤشر الى الحرف o وتغيير الفراغ الى '\n' إذا كانت قيمة p تؤشر الى '' ويستمر هذا طالما أن قيمة p لا تؤشر الى نهاية السلسلة ('\0').

مثال 10-5) المطلوب متابعة البرنامج التالي وما ينتج عنه وقت تنفيذه.

```
#include <iostream.h>
main()
{
	double x,y,*ptr;
	x=123.45;
	ptr=&x;
	y=*ptr;
	cout <<"x ==>"<<x<<"  &x ==>"<<&x<<"  y ==>"<<y
	     <<"  *ptr ==>"<<*ptr<<endl;
	y=*&x;
	cout <<"x ==>"<<x<<"  &x ==>"<<&x<<"  y ==>"<<y
	     <<"  *ptr ==>"<<*ptr<<endl;
	y=x;
	cout <<"x ==>"<<x<<"  &x ==>"<<&x<<"  y ==>"<<y
	     <<"  *ptr ==>"<<*ptr<<endl;
	cout <<"*ptr ==>"<<*ptr<<"  **&ptr ==>"<<**&ptr
	     <<" ptr-(ptr-5) ==>"<<ptr-(ptr-5)<<endl;
}
```

2.10) المؤشرات والدوال Pointers and Functions

في البرامج السابقة وخصوصا عند استخدام الدوال لم يكن في مقدورنا الرجوع الا بقيمة واحدة فقط عند التمرير بالقيمة ، هذه ناحية ، أما من الناحية الاخرى فقد يفرض علينا عند حل المسألة الرجوع باكثر من قيمة واحدة من الدالة الفرعية الى النقطة التي حدثت فيها الاشارة إلى هذه الدالة.

وحتي تكون الدالة قادرة على الحصول على البيانات من المكان الذي تم فيه استدعاؤها وبالتالي اعادة المعلومات الي تلك النقطة، عن طريق المؤشرات pointers ينبغي استخدام :-

1) الرمز(&) وذلك بوضعه قبل المتغير لتوضيح عنوان المتغير في الذاكرة حيث تخزن قيمة المتغير.
2) الرمز (*) الذي يقوم بتتميم لوظيفة الرمز(&) حيث ترجع قيمة المتغير نفسه.

مثال 10-6) خذ مثلا البرنامج التالي :-

```
#include <iostream.h>
pointer(int *);
main()
{
	int a=55;
	cout<<"The value of a ==>:"<<a<<endl;
	cout<<"The address of &a ==>:"<<&a<<endl;
	pointer(&a);
	return 0;
}

pointer(int *ptra)
{
	cout<<"The address of ptra ==>:"<<ptra<<endl;
	cout<<"The value of *ptra ==>:"<<*ptra<<endl;
}
```

في هذا البرنامج مرر عنوان a الى دليل الدالة pointer الذي هو المؤشر المتغير ptra* الذي سيدل على العنوان أو الموقع المخصص للمتغير a في الذاكرة دائما وعليه فان كلا من ptra,a* تكون لهما نفس القيمة ، في حين ان &a يمثل عنوان أو مكان تخزين المتغير a وهو عائد للآلة ، البرنامج السابق سيطبع الآتي عند تنفيذه

```
The value of a ==>:55
The address of &a ==>:0x8e90fff4
The address of ptra ==>:0x8e90fff4
The value of *ptra ==>:55
```

مثال 10-7) دعنا الآن نقوم بكتابة برنامج كامل مهمته مناداة الدالة change_xy التي مهمتها استقبال متغيرين b,a من النوع الحرفي string حيث يتم استبدال محتوى المتغير a بمحتوى المتغير b ، ومحتوي المتغير b بمحتوى المتغير a والرجوع الى نقطة المناداة وطباعة قيمة المتغيرين المتبادلين.

```
#include<iostream.h>
change_xy(a,b)
char a,b;
{
        char c;
        c=*a;
        *a=*b;
        *b= c;
        return(0);
}

main()
{
        char *x="PLEASE";
        char *y="WAIT";
        cout <<"----------------------------";
        cout <<"\nBefore call to function"<<endl
             <<"\tX="<<x<<"   Y="<<y;
```

```
        cout <<"\n--------------------------------";
        change_xy(x,y);
        cout <<"\nAfter call to function"<<endl
             <<"\tX="<<x<<"   Y="<<y;
        cout <<"\n--------------------------------";
        getch();
        return(0);
}
```

وفيما يلي النتائج قبل وبعد الاستبدال :-

```
------------------------------------
 Before call to function
       X= PLEASE   Y=WAIT
------------------------------------
 After call to function
       X= PLEASE   Y=WAIT
------------------------------------
```

بالرغم من أن الادلة جميعها من نفس النوع سواء في الدالة أو في النقطة ألتي حدثت عندها الاشارة الى هذه الدالة ، وبالرغم من تمرير قيمة المتغير xالى المتغيرaوقيمة المتغير y الى المتغير b وبالرغم من عملية التبديل المعروفة التي تمت بينهما عن طريق الجمل

```
c=*a;
*a=*b;
*b=c;
```

فانه عند وصول التحكم الى نهاية الدالة والرجوع الى نقطة الاستدعاء في الدالة الرئيسية ، فقد تمت طباعة قيم المتغيرات x,y خاطئة.

السبب هنا هو أن عملية التمرير من الدالة الفرعية الى الدالة الرئيسية لم تتم لان البرنامج تعامل مع قيم هذه المتغيرات وليس مع عناوينها، وحتى يتم تمرير البيانات من الدالة الرئيسية الى الدالة الفرعية وبالعكس وجب اتباع الآتي :-

1) وضع مؤثر العنوان (&) أمام المتغيرات y,x في نقطة الاستدعاء ليدل على عنوانها.
2) وضع المؤثر (*) امام كل المتغيرات داخل الدالة وهي c,b,a لتصبح مؤشرات تشير الى عناوين هذه المتغيرات.

مثال 10-8) البرنامج التالي هو اعادة للبرنامج بالمثال (10-7) في صورته الصحيحة.

```
#include<iostream.h>
change_xy(a,b)
char *a,*b;
{
      char *c;
      *c=*a;
      *a=*b;
      *b=*c;
      return(0);
}

main()
{
      char *x="PLEASE";
      char *y="WAIT";
      cout <<"----------------------------";
      cout <<"\nBefore call to function"<<endl
           <<"\tX="<<x<<"   Y="<<y;
      cout <<"\n----------------------------";
      change_xy(&x,&y);
      cout <<"\nAfter call to function"<<endl
           <<"\tX="<<x<<"   Y="<<y;
```

```
        cout <<"\n----------------------------";
        getch();
        return(0);
}
```

نلاحظ في هذا البرنامج إن الدالـة change_xy استخدم فـي بدايتهـا المؤشـر (*) مـع الادلـة الصوريـة b,a التـي هـي مـن النـوع الحرفـي واسـتعمل مؤشـر العنوان (&) مع المتغيرات الفعلية y , x وبالتالي تم ارسال القيمتيـن وتبديلهمـا عن طريق هذه الدالة والرجوع بهما إلى نقطة الاستدعاء ، وفيما يلـي النتـائج قبل وبعد عملية الاستبدال :-

```
-----------------------------
 Before call to function
        X= PLEASE   Y=WAIT
-----------------------------
After call to function
        X= WAIT   Y= PLEASE
-----------------------------
```

عموما فانه اذا وقـع أي تغيير فـي الادلـة الصوريـة بالدالـة الفرعيـة عنـد استعمالنا للمؤشـرات فسـوف يقابلـه نفس التغيير فـي الادلـة الفعليـة بالدالـــة الرئيسـية ، وينبغـي ان تكـون الادلـة الفعليـــه متغـيرات فقـط ، أي لا يمكـن استخدام الثوابت في هذه الحالة.

مثال 10-9) اوجد ناتج البرنامج التالي:-

```
#include <iostream.h>
main()
{
        void    func( const float * );
        float a=7;
```

```
        func(&a);
        cout<<"ansr="<<a;
        getch();
        return 0;
}

void func(  const float *ptr)
{
        *ptr*=*ptr;
}
```

نلاحظ أنه قد تم الاعلان في الدالة func عن المتغير الثابت ptr الذي هو مؤشر إلى قيمة من النوع الحقيقي ، ومهمة هذه الدالة هي تربيع قيمة هذا المتغير والرجوع بها الي نقطة الاستدعاء ، ولكن هذا لن يحدث والسبب أن ptr هو ثابت أي لايمكن التغيير في قيمته كما ذكرنا سابقا وبالتالي فان البرنامج سيصدر الرسالة التالية:-

Cannot modify a const object

وللحصول على النتائج صحيحة يجب الغاء كلمة الثابت const من تعريف الدالة لتصبح

```
void func(float *ptr)
```

وكذلك عينتها لتصبح

```
void    func(float * );
```

مثال 10-10) البرنامج التالي يقوم بقراءة ثلاث متغيرات من النوع الصحيح ومن ثم استدعاء دالة فرعية مهمتها اعادة ترتيب قيم هذه المتغيرات تصاعديا وقت الضرورة باستخدام الادلة المؤشرة.

```
#include<iostream.h>
main()
{
	void change(int *,int *);
	int a,b,c;
	cout<<"arrange three data items"<<"\n";
	cout<<"************************"<<"\n";
	cout<<"\n"<<"Type three data items ==>:";
	cin>>a>>b>>c;
	if(a>b)
		change(&a,&b);
	if(b>c)
	{
		change(&b,&c);
		if(a>b)
			change(&a,&b);
	}
	cout <<"Data in ascending order :";
	cout <<" "<<a<<" "<<b<<" "<<c<<"\n";
	return 0;
}

void  change(int *i, int *j)
{
	int temp=*i;
	*i=*j;
	*j=temp;
	return ;
}
```

هذا البرنامج سيطبع الثلاثة قيم المدخلة بترتيب تصاعدي وذلك لجميع الاحتمالات ، حيث تم استدعاء الدالة change() لاكثر من مرة والتي مهمتها تبديل قيم المتغيرات وقت الحاجة، وفي كل مرة استخدمت المؤشرات عند استدعائها وذلك بوضع المؤشر (&) امام المتغيرات الفعلية المستعملة b,a والمؤثر (*) امام المتغيرات الصورية j, i حتى يتم تحويلها الى مؤشرات.

فمثلا

```
change(&a,&b);
```

تعني أن كلا من b,a قد اصبحتا مؤشرين .

أما فيما يخص الدالة التي كانت بالشكل التالي :-

```
change(int *i,int *j)
```

تم الاعلان عن مؤشرين إلى قيمتين من النوع الصحيح i , j وهما مؤشران يشيران الى عناوين المتغيرين j,i وبالتالي فانه ما يحدث للمتغير i في الدالة يقابله نفس التغيير بالمتغير a في نقطة الاستدعاء ونفس الشيء يقع بين المتغيرين j و b. تنفيذ البرنامج وادخال البيانات تبقى مهمة القارئ.

3.10) المراجع والدوال References and Functions

المرجع (Reference) يستخدم في الاشارة الى قيمة معينة في ذاكرة الحاسب وهو يماثل عمل المؤشر ولكنه اسهل منه في تمرير قيم الادلة الفعلية من الدالة الرئيسية الى الادلة الصورية بالدالة الفرعية وبالعكس ، أي أن كل الادلة الفعلية يقابلها نفس موقع التخزين للادلة الصورية ، وأي تغيير في الادلة الصورية تنعكس على الادلة الفعلية .

مثال 10-11) البرنامج التالي هو اعادة للبرنامج بالمثال (10-10) السابق باستخدام الدالة الخطية inline والاستدعاء بالمرجع (Call by reference).

```
#include <iostream.h>
#define STR "Data in ascending order ==>: "
int swap(int &, int & );
inline int swap(int &x ,int &y)
```

```
{
        int z;
        z=x;
        x=y;
        y=z;
        return 0;
}

void main()
{
        int i,a,b,c,z;
        cout <<" Arrange three data items"<<endl
              <<"------------------------"<<endl;
        cout <<endl<<"Type 3 data items ==>: ";
        cin>>a>>b>>c;
        if(a>b)
              swap(a,b);
        if(b>c)
              swap(b,c);
        if(a>b)
              swap(a,b);
        cout<<STR<<' '<<a<<' '<<b<<' '<<c;
}
```

كما نلاحظ في هذا البرنامج القيام باستخدام عينة الدالة swap من النوع الصحيح بالشكل

```
int swap(int &, int & );
```

لتدل على ان دليليها مرجعان من النوع الصحيح وذلك باستخدام مؤثر العنوان (&) ويطلق على هذا النوع من الاستخدام استدعاء الدالة بالمرجع (call-by-reference) ، البرنامج وقت التنفيذ وادخال البيانات سيعطي نفس النتيجة بالبرنامج المشار اليه بالمثال .

مثال 10-12) المطلوب شرح البرنامج التالي :-

```
#include <iostream.h>
void print_out(int& x, int y, int& z);
main()
{
	int x=3,y=4,z=5;
	cout<<"\t\t\tX  \t Y  \t Z"<<endl;
	print_out(x,y,z);
	cout <<"After return "
		<<"\t\t"<<x<<" \t"<<y<<"\t"<<z<<endl;
	return 0;
}

void print_out(int &x, int y, int &z)
{
	cout <<"Function begin "
		<<"\t\t"<<x<<"\t"<<y<<"\t"<<z<<endl;
	x=300;
	y=400;
	z=500;
	cout <<"Function end "
		<<"\t\t"<<x<<"\t"<<y<<"\t"<<z<<endl;
}
```

وقت التنفيذ ينتج التالي :-

	X	Y	Z
Function begin	3	4	5
Function end	300	400	500
After return	300	4	500

ما نلاحظه في هذا البرنامج وباختصار أن كل ما يحدث للمتغيرين x , z بالدالة print_out يقابله نفس التغيير بالمتغيرين x , z بالدالة الرئيسية، وبالتالي تغيرت قيمتهما عند الرجوع والسبب هو وضع المؤثر (&) مع كل واحدا منهما في بداية الدالة ، أما المتغير y لم يرجع بالقيمة الجديدة المتحصل عليها بالدالة وبقيت قيمته على ما هي لانه لم يستخدم المؤثر & معه.

مثال 10-13) المطلوب كتابة دالة رئيسية تقوم باســتقبال سلسلة حرفيـة string ومن ثم استدعاء :-

1) دالة المعاودة الذاتية Recursion Function ووظيفتها تحديد طول السلسلة.
2) دالة اخرى مهمتها استقبال السلسلة وطباعة حروفها .

```
#include <iostream.h>
int str_length(char *str);
void print_str(const char *S);
void main()
{
	char *k ="Hi Welcome to computer dept.";
	int j = str_length(k);
	cout<<"\n the string is:";
	print_str(k);
	cout<<"\n this string has ["<<j<<"] characters.";
	getch();
}

int str_length(char *str)
{
	if(*str)
		return (1+str_length(str+1));
	else
		return 0;
}

void print_str (const char *S )
{
	for( ; *S != '\0' ; S++ )
	cout<<*S;
}
```

بدأ البرنـامج باسـناد سلسلة حرفيـة للمتغير k ثـم اسـتدعاء دالـة الاعـادة str_length التي مهمتها استقبال السلسلة الحرفية وبالتالي تخزينها في المؤشر

str* فاذا كان شرط جملة if غير صحيح أي كـانت السلسلة فارغة ، عندها ترجع الدالة بالقيمة 0 الناتجة من الجملة

```
return 0;
```

أما اذا كانت غير ذلك تنفذ الجملة التالية

```
return(1+str_length(str+1));
```

وتعني اضافة القيمة 1 الى قيمة الدالة str_length() وفي نفس الوقت تستدعي الدالة نفسها على ان تزداد قيمة المؤشر بمقدار 1عن طريق التعبير (str+1)، ويتكرر هذا حتي يصل المؤشر الى الرمز ('\0') الذي يدل على نهاية السلسلة ، عندها ترجع الدالة بقيمة واحدة من النوع الصحيح int والذي يمثل طولها الى نقطة الاستدعاء.

اما فيما يخص دالة الطباعة print_str التي تحتوي على دليل واحد وهو المتغير الثابت من نوع المؤشر الحرفي S الذي تمرر عبره السلسلة الحرفية المراد طباعتها ، كما تحتوي الدالة على جملة for والتي مهمتها طباعة السلسة حرفا حرفا حتي نهايتها باستخدام المؤشر S.

البرنامج يعطي الناتج الآتي وقت التنفيذ

```
Type your string : Hi Welcome to computer dept.
The string is : Hi Welcome to computer dept.
This string has [28] characters.
```

وحتى نستطيع تلخيص العلاقة بين المتغيرات المحلية والخارجية والساكنة والمؤشرات واستعمالها مع عدد من الدوال الفرعية، المثال التالي يوضح ذلك

مثال 14-10) اكتب برنامجا كاملا لادخال نتيجة الامتحان الاول والثاني في مادة الحاسب لكل طالب في فصل به عدد number من الطلبه وكتابة دالة فرعية اولى مهمتها :-

a) ايجاد اكبر درجة امتحان مع ايجاد المتوسط لكل طالب.

b) الافادة بان الطالب ناجح (Pass) اذا كان المتوسط اكبر من او يساوى 50 اوغير ناجح (Fail) اذا كان المتوسط غير ذلك.

مع طباعة كل البيانات والمعلومات السابقة عن طريق دالة فرعية ثانية.

لايجاد المطلوب ينبغي كتابة الدالة الرئيسية التي مهمتها ادخال البيانات الخاصة لطلبة الفصل ومناداة عدد من الدوال وقد تكون بالشكل الموالي:-

```
#include <iostream.h>
float avg;
char *status;
main()
{
        void calculate(const float,const float,float *);
        void print_them(const float , const float,const float *);
        int no;
        float max, t1,t2;
        cout<<"Give number of students and their tets:"<<"\n";
        cin>>no;
        cout <<"\n number test1  test2   max  avg  status "
             <<"\n---------------------------------------\n";
        for(int i=1;i<=no;i++)
        {
             cin>>t1>>t2;
             calculate(t1,t2,&max);
             print_them(t1,t2,&max);
        }
        getch();
        return 0;
}
```

في هذه الدالة تم الاعلان عن المتغير status وهو مؤشر إلى قيمة من النوع الحرفي وذلك قبل بداية الدالة الرئيسية main() وبالتالي فهو متغير خارجي لاستعماله في الدالة الرئيسية وبقية الدوال التابعة لها.

بعد ادخال البيانات عن طريق جملة for التي تخص الطالب الاول تم استدعاء الدالتين التاليتين:

1) الدالة الاولي اسمها calculate() وهي

```
void calculate(const float m1, const float m2,float *max)
{
        float sum;
        if(m1>m2)
              *max=m1;
        else
              *max=m2;
        sum=m1+m2;
        avg=sum/2;
        if(avg>=50)
              status="Pass";
        else
              status="Fail";
        return;
}
```

حيث مررت لها قيمة الامتحان الاول والثاني للطالب الاول من الدالة الرئيسية، ومن ثم جري ايجاد اكبر امتحان وتخصيصه للمتغير max من نوع المؤشر، ايضا وقع تخصيص حالة الطالب ناجح أو غير ناجح بناء على متوسط درجاته للمتغير الخارجي status واخيرا الرجوع بهذه المعلومات الى مكان الاستدعاء.

2) الدالة الثانية والتي اسمها print_them() مررت لها البيانات والمعلومات من الدالة الرئيسية عن طريق المؤشرات من النوع الثابت مع الاعلان عن

المتغير الساكن counter من النوع الصحيح واعطيت له القيمة المبدئية 1 حيث تمت طباعة البيانات المدخلة والمعلومات الناتجـة من الدالـة الفرعيـة السـابقة calculate والتي تخص الطالب رقم 1 وهي:

counter متغير ساكن يستخدم لتسلسل رقم الطالب.

n1,n2 ثابتان بهما درجة الطالب الاول.

max متغير محلي يمثل اكبر درجة.

avg متغير خارجي يمثل متوسط الدرجات .

status متغير خارجي وهو يمثل حالة طالب اما ناجح أو غير ناجح.

ثم تزداد قيمة المتغير الساكن counter بالعدد 1 فـاصبحت 2 وعنـد استدعاء هذه الدالة للمرة الثانية تطبع البيانات والمعلومات التي تخص الطالب رقم 2 ، وهكذا تتكرر نفس العملية حتى نهايـة جملـة for فـي الدالـة الرئيسـية ، وهـذه الدالة هي :

```
void print_them(const float n1, const float n2,const float *max)
{
        static int counter=1;
        cout <<"\n"<<counter<<"\t"<<n1<<"\t"<<n2<<"\t"
             <<*max<<"\t"<<avg<<"\t"<<status<<"\n";
        counter++;
        return;
}
```

أخيرا وحتى يكون العمـل مكتمـلا ، فينبغـي ربـط الـدوال الفرعيـة بالدالـة الرئيسية ، واذا ما تم ذلك وادخلت البيانات المطلوبة ، فسـوف تكـون النتـائج مشابهة للآتي :-

```
Give number of students and their tests:
5    65    35    45    48    90    86    50    51    72    71

number    test1    test2    max    avg    status
------------------------------------------------------------
  1        65       35      65     50     Pass
  2        45       48      48     46.5   Fail
  3        90       86      90     88     Pass
  4        50       51      51     50.5   Pass
  5        72       71      72     71.5   Pass
```

4.10) المؤشرات والمصفوفات ذات البعد الواحد

Pointers and One-dimensional Arrays

يتم التعامل بين المؤشرات والمصفوفات وكذلك الدوال بجميع أنواعها في حل الكثير من المسائل كبيرة الحجم ، وتتعامل السلاسل على انها نوع من تسلسل عدد من الرموز حيث يؤشر المترجم الى الرمز الاول منها أي يمكن ان نقول ان المؤشر يؤشر الى بداية اية سلسلة حرفية .

مثال 10-15) خذ مثلا الاعلان

```
char *name[4] = {"My name is","BASHIR","ALI","GAYED"};
```

يعني ان المصفوفة name [4] هي مؤشر من النوع الحرفي لها عدد 4 من العناصر كل واحدة من هذه العناصر تخزن في ذاكرة الحاسب بعدد طول السلسلة المحصورة بين علامتي التنصيص المزدوجة مضافا اليها مكان نهايتها ، أي ان اطوال السلاسل الاربعة تكون 6,4,7,11 على التوالي، حيث تعطي لغة سى++ المبرمج القوة لانجاز ومعالجة السلاسل بأي طول يختاره، والشكل التالي يبين عناصر المصفوفة name السابقة.

```
               |-------------------------------------------|
name[0]---->|  'M' 'y' ' ' 'n' 'a' 'm' 'e' ' ' 'i' 's' '\0'  |
               |-------------------------------------------|
name[1]---->|   'B' 'A' 'S' 'H' 'I' 'R' '\0'                |
               |-------------------------------------------|
name[2]---->|   'A' 'L' 'I' '\0'                            |
               |-------------------------------------------|
name[3]---->|   'G' 'A' 'Y' 'E' 'D' '\0'                    |
               |-------------------------------------------|
```

حيث نلاحظ في نهايــة كل سلسـلة رمـز النهايـة ('\0') لكـي تـأخذ السلسـلة اقصر طول لها.

مثال 10-16) البرنامج التالي يبين استخدام المصفوفة من نوع المؤشر

```
#include <iostream.h>
main()
{
      char *name[4] = {"My name is","BASHIR","ALI","GAYED"};
      for(int i=0;i<4;i++)
            cout<<"name["<<i<<"] ==>"<<name[i]<<endl;
      getch();
      return 0;
}
```

وينتج عنه الآتي :-

```
name[0] ==>My name is
name[1] ==>BASHIR
name[2] ==>ALI
name[3] ==>GAYED
```

مثال 10-17) خذ مثلا الاعلان

```
int mat[10];
```

يدل على انه تم الاشهار عن مصفوفة احادية من النوع الصحيح حجمها 10 عناصر ، حيث يمكن أن نؤشر الى أول عنصر في هذه المصفوفة بالطريقة اسم المصفوفة mat أو بالطريقة &mat[0]

مثال 10-18) جملة التخصيص

```
min=mat[2];
```

حيث استخدم الدليل 2 ليشير الى قيمة العنصر الثالث في المصفوفة mat وبالتالي جري اسناد تلك القيمة الى المتغير min وهذه الجملة لها نفس تأثير الجملة

```
min=*(mat+2);
```

حيث استخدم المؤشر الذي يشير الى العنصر mat[0] مضافا اليه العدد2 وهو يقابل العنصر الثالث في نفس المصفوفة.

ملاحظة: يجب أن نأخذ في الاعتبار أن يكون وضع الرمز (*) قبل الأقواس ، لأن الأقواس لها افضلية أعلى من الرمز (*).

مثال 10-19) في البرنامج الموالي يتم قراءة عدد من القيم الصحيحة وتخزينها في مصفوفة ذات بعد واحد ثم ايجاد وطباعة أصغر قيمة عنصر مع تحديد مكانها في هذه المصفوفة.

```
// This program determines and prints the smallest value in an array
#include <iostream.h>
const int LEN= 5;
void main()
```

```
{
      int mat[LEN],i,min,pos,*ptr;
      ptr=mat;
      cout<<"Enter elements of an array :"<<endl;
      for(i=0;i<LEN;i++)
      {
            cout<<"Enter MAT["<<i<<"] ==>:";
            cin>>mat[i];
      }
      min=*ptr;
      for(i=0;i<LEN;i++)
      if(*(ptr+i)<=min)
      {
            min=*(ptr+i);
            pos=i;
      }
      cout <<endl<<"The smallest element is "
            <<"MAT["<<pos<<"] ==>: "<<min;
      getch();
}
```

تم الاعلان عن مصفوفة mat احادية من النوع الصحيح ، مع الاعلان عن المتغير المؤشر ptr ، يلي ذلك الجملة

```
ptr=mat;
```

التي تعني اسناد عنوان mat[0] الى المؤشر ptr ، أو بمعني آخر ptr يشير الى mat[0] .

بعد ادخال القيم اللازمة وتخزينها في المصفوفة ، وبما أن المطلوب ايجاد أصغر قيمة عنصر في هذه المصفوفة ، جاء الأمر التالي :-

```
min=*ptr;
```

وهي تخصيص أول عنصر وهو الذي يؤشر اليه المؤشر ptr الى المتغير الصحيح min ، وعلى هذا بدأت جملة for البحث عن اصغر عنصر بداية من

العنصر الثاني بحيث اجريت المقارنة بين ثاني عنصر في المصفوفة مع قيمة المتغير min باستخدام جملة if كالتالي :-

```
if( *(ptr+i) < min )
```

في كل مرة تزداد قيمة العداد i الى المؤشر ptr وتقارن مع قيمة المتغير min فاذا كان الشرط صحيحا (true) حينها يتم تخصيص القيمة الصغري للمتغير min باستخدام المؤشر اما بالطريقة

```
min=*(ptr+i);
```

أو بالطريقة

```
min=*(&ptr[0]+i);
```

وفي نفس الوقت يخصص قيمة الدليل i للمتغير pos لتحديد مكان القيمة الصغرى ، وهكذا حتي يتحقق شرط جملة for ، تنفيذ هذا البرنامج سيعطي الرسالة التوضيحية الدالة على ادخال قيم عناصر المصفوفة كالآتي :-

```
Enter elements of an array :
Enter MAT[0] ==>: 8
Enter MAT[1] ==>: 5
Enter MAT[2] ==>: 1
Enter MAT[3] ==>: 2
Enter MAT[4] ==>: 4
```

يلي هذا الحصول على القيمة الصغرى في المصفوفة ومكان وجودها.

```
The smallest element is MAT [2] ==>:1
```

مثال 10-20) هذا برنامج آخر يوضح العلاقة بين المؤشرات والمصفوفة أحادية البعد.

```
#include <iostream.h>
main()
{
        int mat[]={5,9,4,1,6,8,0,2,7,3};
        int *ptr;
        ptr=mat;
        for (int i=2;i<=8;i++)
        cout<<*ptr++<<"  ";
}
```

في هذا البرنامج وبعد تخصيص القيم المبدئية للمصفوفة mat ثم الاعلان عن المتغير ptr من نوع المؤشر جاءت بعدها جملة التخصيص ptr=mat وبالتالي تمت طباعة القيم باستخدام جملة for وعلى القارئ استنتاج ناتج هذا البرنامج.

5.10) المؤشرات والمصفوفات ذات البعدين

Pointers and Tow-dimensional Arrays

مثال 10-21) الاعلان التالي

```
int array1[10][10];
```

يعني الاشهار عن مصفوفة ذات بعدين تحت اسم array1 من النوع الصحيح ، وعليه يمكن استخدام العنوان الأساسي للمصفوفة اما عن طريق اسم المصفوفة array1 أو عن طريق &array1[0][0] لتحديد بداية عنوان هذه المصفوفة ، حيث تم حجز 100 موقع في ذاكرة الحاسب للمصفوفة المعنية.

مثال 10-22) الجملة

```
array2[i][j]=array1[i][j];
```

تعني تخصيص القيمة الموجودة في الصف i والعمود j من المصفوفة array1 الى نفس الصف والعمود بالمصفوفة array2.

يمكن عمل التخصيص السابق باستخدام المؤشرات بعدة طرق منها :

1) `array2[i][j]=*(array1[i]+j);`

2) `array2[i][j]=(*(array1+i))[j];`

3) `array2[i][j]=*(&array1[0][0]+10*i+j)`

الرقم 10 هنا يمثل حجم عمود المصفوفة.

مثال 10-23) البرنامج التالي يتم فيه قراءة عدد من القيم الصحيحة وتخزينها باتجاه العمود في مصفوفة ذات بعدين ، ثم اخراج هذه القيم باتجاه الصف مع ايجاد مجموع قيم عناصر هذه مصفوفة.

```
#include <iostream.h>
const int ROW=2, COL=2;
main()
{
      int x[ROW][COL],i,j, sum=0;
      cout<<"Enter an array :\n";
      for(j=0;j<COL;j++)
      for(i=0;i<ROW;i++)
      {
           cout<<"Enter X["<<i<<","<<j<<"] ==>: ";
           cin>>x[i][j];
           sum+=x[i][j];
      }
      cout<<"Here is the array :\n";
      for(j=0;j<ROW;j++)
      {
           for(i=0;i<COL;i++)
                cout<<(*(x+j))[i]<<"\t";
           cout<<endl<<endl;
      }
      cout<<"the sum is ==>"<<sum;
      return 0;
}
```

يمكن استخدام المؤشر مع جملة الادخال بالطريقة التالية :-

```
cin>> (*(x+j))[i];
```

كما هو معمول به مع جملة الطباعة كالآتي :-

```
cout<< (*(x+j))[i];
```

وقد تكتب هذه الجملة كالآتي :-

```
cout<< *(&x[0][0]+COL*j+i);
```

أو تكتب بالطريقة المعتادة

```
cout<< x[j][i];
```

عند تنفيذ هذا البرنامج وادخال قيم هذه المصفوفة

```
Enter an array :
Enter X[0,0] ==>: 3
Enter X[1,0] ==>: 4
Enter X[0,1] ==>: 5
Enter X[1,1] ==>: 6
```

حيث تم تخزين القيم باتجاه العمود column-wise واخراج نفس القيم ولكن باتجاه الصف row-wise مع المجموع على النحو التالي :-

```
Here is the array :
   3    5
   4    6
the sum is ==> 18
```

التعامل أو الاتصال بين الدالة الرئيسية main() والدوال الفرعية التابعة لها عند معالجة بيانات كثيرة جدا باستخدام المصفوفات ، قد يتم باستخدام المؤشرات عند تمرير هذه البيانات بينها جميعا .

مثال 10-24) المطلوب كتابة برنامج كامل لمعالجة بيانات تخص طلبة في فصل لايزيد عددهم على 10 طلبة ، على ان يتم تخزين البيانات على هيئة مصفوفتين ، الأولى تحتوي عناصرها على درجات قديمة لمقرر الحاسب الآلي ، أما الثانية فهي مصفوفة جديدة تستقبل وتخزن درجات لعدد N من الطلبة لهذا الفصل ، المطلوب طباعة درجات أي من المقررين بترتيب تصاعدي مع طباعة متوسط هذه الدرجات في كل حالة.

وحتى يسهل فهم ومتابعة البرنامج ، وبالتالي تمرير قيم احد المصفوفتين باستخدام المؤشرات ، ينبغي أن تكتب الدالة الرئيسية أولا ، ثم الدوال المعنية كلا على حده.

```
#include <iostream.h>
#include <conio.h>
const int MAX= 10;
main()
{
        void insert_data(const int k, int NewGrade[]);
        void sort_data(int *p, const int n);
                        \\ Grade for old course
        int OldGrade[]={68,45,50,70,69,77,62,91,69,31};
        int i,op,n, NewGrade[MAX];
        for(;;)
        {
            clrscr();
            cout <<endl<<"\tEnter"
                <<endl<<"\t  1))To insert grade for new course"
                <<endl<<"\t  2))To get grade for old course"
                <<endl<<"\t  3))To exit"
                <<endl<<endl<<"\t\tYour choice please ==>: ";
```

```
          cin>>op;
          clrscr();
          switch(op)
          {
                case 1:  cout<<endl<<"Enter class size ==>: ";
                         cin>>n;
                         if ((n>10) || (N<=0))
                              exit (0);
                         insert_data(n, NewGrade);
                         sort_data(NewGrade,n);
                         break;
                case 2:  sort_data(OldGrade,MAX);
                         break;
                case 3:  exit(0);
                         break ;
          }
     }
}
```

تم الاعلان في بداية الدالة الرئيسية عن المصفوفه OldGrade وهي غير محددة الطول وبها درجات الطلبة القديمة في مقرر الحاسب الآلي على هيئة قيم ابتدائية ومصفوفة أخرى NewGrade لاستقبال وتخزين N من الدرجات في اي مقرر آخر كما تحتوي هذه الدالة على قائمة بالاختيارات التالية:-

1) ادخال عدد الطلبة ودرجاتهم بالفصل في المقرر الجديد عن طريق الدالة الفرعية insert_data مع ترتيب هذه الدرجات تصاعديا وحساب وطباعة متوسط الفصل عن طريق الدالة الفرعية sort_data ويتم تنفيذ هذا المطلوب بالاختيار رقم 1.

2) طباعة درجات الطلبة في المقرر القديم في ترتيب تصاعدي مع حساب وطباعة متوسط هذه الدرجات عن طريق الدالة الفرعية sort_data ويتم هذا بالاختيار رقم 2.

3) الخروج نهائيا من البرنامج ، ويتم هذا بالاختيار رقم 3 وذلك باستخدام جملة switch حيث تستدعى الدالة عن طريق ادخال رقم الاختيار.

الدالة الفرعية insert_data مهمتها استقبال درجات لعدد k من الطلبة في المقرر الجديد وتخزينها في مصفوفة NewGrade والرجوع بها الى الدالة الرئيسية لمعالجتها وهي:

```
        //  insert new grades function
void insert_data(const int k, int NewGrade[])
{
        int i;
        clrscr();
        cout<<"\n Enter "<<k<<" new grades\n";
        for(i=0;i<k;i++)
        cin>> NewGrade[i];
}
```

أما الدالة االفرعية الأخيرة وهي sort_data فمهمتها استقبال عدد الطلبة بالفصل عن طريق دليلها الثابت الصحيح n والدرجات عن طريق الدليل الأول p* الذى هو مؤشر إلى قيم من النوع الصحيح، لاى من المصفوفتين OldGrade أو NewGrade، وتقوم بترتيب قيم عناصر المصفوفة ترتيبا تصاعديا عن طريق استدعاء الدالة الفرعية swap وذلك في حالة ما اذا كانت قيمة دليل العنصر الاول من المصفوفة اكبر من العنصر الموالي له باستخدام دالة التبديل swap وهي:

```
void swap(int *elementp1, int *elementp2)
{
        int   temp=*elementp1;
        *elementp1=*elementp2;
        *elementp2=temp;
}
```

حيث تستخدم عملية تبديل قيم العنصرين عـن طريـق متغير ثـالث مؤقت temp والرجوع الى الدالة السابقة حيث تطبع الدرجات مرتبة مـع متوسـطها وهذه الدالة هي:

```
//   sort grades function
void sort_data(int *p, const int n)
{
        void swap(int *, int *);
        int i,j,temp;
        float avg,sum=0.0;
        clrscr();
        for(i=1;i<n;i++)
        for(j=0;j<n-1;j++)
        if(p[j]>p[j+1])
                    //  Call swap function
        swap(&p[j], &p[j+1]);
        cout <<"\t The grade after sorting\n"
              <<"\t-----------------------\n";
        for(i=0;i<n;i++)
        {
              cout<<"\n grade ["<<i+1<<"] = "<<*(p+i);
              sum+=*(p+i);
        }
        avg=sum/n;
        cout <<"\n The exam average = "<<avg
              <<"\n Press any key to continue";
        getch();
}
```

عموما فعند ربط الدالة الرئيسية مع بقيـة الـدوال، وتنفيذ البرنـامج تمسـح شاشة العرض ويحدث نوع من التحاور بين الحاسب الآلـي والمسـتعمل لهـذا البرنامج ، ويبدأ بإدخال الاختيار المطلوب من ضمن الاختيارات التالية :-

```
Enter
    1))To insert grade for new course
    2))To get grade for old course
    3))To exit
   Your choice please ==>:
```

واذا ما تم اختيار رقم 2 كالآتي :-

```
Your choice please ==>: 2
```

فسيتم اعادة ترتيب الدرجات بالمصفوفة القديمة OldGrade مع المتوسط على النحو التالي :-

```
The grade after sorting
-----------------------------------
grade[1]=31
grade[2]=45
grade[3]=50
grade[4]=62
grade[5]=68
grade[6]=69
grade[7]=69
grade[8]=70
grade[9]=77
grade[10]=91

The exam average = 63.200001
```

يلي ذلك الرد على السؤال

```
Press any key to continue
```

بالضغط على أي مفتاح في لوحة المفاتيح للاستمرار ، واذا ما تم ذلك تظهر على الشاشة مرة اخرى قائمة بالاختيارات

Enter

1))To insert grade for new course

2))To get grade for old course

3))To exit

Your choice please ==>:

فاذا ما اختير الاختيار الاول

Your choice please ==>: 1

يطلب البرنامج عدد الطلبة بالفصل وليكن 5

Enter class size ==>: 5

في هذه الحالة يأتي السؤال بادخال عدد 5 درجات

Enter 5 new grades

76 41 90 68 59

ومن ثم يظهر ترتيب هذه الدرجات تصاعديا مع متوسطهم

The grades after sorting

----

grade[1]=41

grade[2]=59

grade[3]=68

grade[4]=76

grade[5]=90

The exam average = 66.800001

Press any key to continue

وهكذا تستمر عملية تنفيذ البرنامج والدوال التابعة له حتي يتم ادخال الرقم 3 لانهاء عملية التنفيذ.

6.10) تمارين Exercises

استخدم المؤشرات كلما امكن ذلك عند كتابة أي برنامج في هذا التمارين.

(1) اذا اعطيت الاشهارين

```
char string[25];
char *str;
```

ما الفرق بين الاشهارين ؟ وكم حرفا يمكن أن يخزن في كل واحد منهما؟ ولماذا ؟

(2) اذا اعطيت الاعلان والتخصيص التالي

```
static int x[6]={11,22,33,44,55,66};
```

المطلوب ايجاد القيمة التي تطبع في الآتي :-

```
(a) cout<<*x<<endl;
(b) cout<<(*x+2)<<endl;
(c) cout<<*(x+4)<<endl;
(d) cout<<(*(x)+5)<<endl;
```

(3) اكتب برنامجا كاملا لقراءة درجة الطالب واسمه لفصل به عدد num من الطلبة ، بعد تخزين هذه البيانات في مصفوفات مناسبة ، اطبع قائمة باسماء الطلبة الذين لهم الاسم الاول ALI مع درجاتهم .

(4) اكتب برنامجا مهمته استدعاء دالة تستقبل اربعة قيم من النوع الحرفي والرجوع باكبر واصغر هذه القيم.

(5) اذا اعطيت التعريف التالي :-

```
char str[]="The capital with the fat"
```

المطلوب كتابة برنامج مهمته معالجة السلسلة في التعريف السابق وتحويلها بالصورة:

The cat with the rat

(6) اكتب برنامجا باستخدام المؤشرات لقراءة سلسلة حرفية لا يزيد طولها على 25 حرفا أو رمزا مع ترتيبها تصاعديا.

(7) اوجد ناتج تنفيذ البرامج الموالية

(a)

```
#include<iostream.h>
main()
{
	int x=35;
	int &y=x;
	int *p=&x;
	x++;
	cout<<"X="<<x<<" Y="<<y<<"  *P="<<*p<<endl;
	y++;
	cout<<"X="<<x<<" Y="<<y<<"  *P="<<*p;
}
```

(b)

```
#include <iostream.h>
int get_result(char *, char *);
main()
{
	char str1[10]="12345",str2[10]="ABCDE";
	get_result(str1,str2);
	cout<<" The result is "<<str1<<endl;
	return 0;
}

int get_result(char *s1,char *s2)
{
	while(*s1 !='\0')
```

```
        {
            cout<<*s1;
            ++s1;
        }
        for( ; *s1=*s2;s1++,s2++)
        ;
}
```

(c)

```
#include <iostream.h>
const int MAX= 10;
main()
{
        void fun_one(int n,int arr[]);
        void fun_two(int *p,int m);
        int mat[]={7,4,11,20,25,71,16,21,18,27};
        cout<<"First output is :"<<endl;
        fun_one(MAX,mat);
        cout<<endl<<"Second output is :"<<endl;
        fun_two(mat,MAX);
        for(int i=0;i<MAX;i++)
        cout<<mat[i]<<"   ";
}

void fun_one(int size, int matr[])
{
        int i;
        for(i=0;i<size;i++)
        if(i % 2 ==0)
        cout<<matr[i]<<"   ";
}

void fun_two(int *array,int n)
{
        for(int i=0;i<n;i++)
        if(*(array+i) % 2==1)
        (*(array+i))++;
}
```

(8) المطلوب قراءة اسم الطـالب RAEF BASHIR ودرجتـه 83.5 ثـم اخـراج هذه البيانات بالشكل التالي على شاشة العرض.

Student Name ##### RAEF BASHIR ##### has grade 83.5

(9) أكتب برنامجا كاملا لقراءة مصفوفة صحيحـة ، ثـم استخدم دالـة واحـدة لايجاد وأعادة مجموع كل العناصر الموجبة فـي المصفوفـة عـن طريـق جملة return. وعدد العناصر الســالبة فـي المصفوفـة مـن خـلال المتغـير الخارجي واخيراً أكبر عنصر في المصفوفة من خــلال متغير مـن نـوع مؤشر.

(10) أكتب برنامجا لقراءة سلسلة حرفية لا تزيد عن 50 حرفاً حيث تنتهي هـذه السلسلة بالرمز (*) ، أطبع هذه السلسلة عكسيا ، فمثلا :

اذا كانت المدخلات 91+c+b/5=a

فالاخراج يكون a=5/b+c+19

(11) باستخدام المؤشرات اعد كتابة تمرين (14) بالفصل السادس.

(12) اكتب برنامجا يستدعي دالة مهمتها ايجاد عدد مـرات تكـرار الرمـز (*) في سلسلة يتم ادخالها عن طريق دالة أخرى.

(13) المطلوب كتابة برنامج يقوم باستدعاء الدالة

```
void DigitEvenSum(int n, int *Psum)
```

التي مهمتها الرجوع بمجمـوع الاعداد الزوجيـة للمتغير n عن طريـق المؤشر Psum*.

الفصل الحادي عشر: تراكيب البيانات

الهدف من هذا الفصل هو التطرق الى ذكر بعض التعريفات الجديدة للبيانات وكيفة استغلال ذاكرة الحاسب لتخزين أكثر من متغير في نفس المكان، ايضا شرح طريقة التعامل مع تراكيب البيانات (Data Structures) التي تشتهر بها لغة سي++ مع الدوال أو مع المصفوفات.

1.11) استخدام انواع جديدة Typedef

في الفصول الماضية تم الاعلان عن معرفات داخل البرنامج الواحد بقصد تخزين بيانات مختلفة فيها مثل الصحيح int والحقيقي float والحرفيchar وغيرها ، بالاضافة الى ذلك فان هذه اللغة تمكننا من استحداث تعريفات جديدة للبيانات عن طريق الكلمة typedef وهي لا تسمح بتعريف نوع جديد من البيانات، ولكن بتعريف مكافيء لنوع موجود أصلا.

الشكل العام

```
typedef  type new-name-of-type;
```

حيث:

typedef كلمة محجوزة مهمتها السماح بتحديد النوع الجديد .

type النوع.

new-name-of-type الاسم الجديد المطلوب استخدامه .

مثال 11-1) خذ مثلا التعريف

```
typedef  int NUMBER;
```

يعني أن الاسم NUMBER الذي كتب بحروف كبيرة حتى يكون مختلفا عند استخدامه في الاعلان لتحديد أنواع اخرى مشتقة من هذا الاسم كما يلي:-

```
NUMBER num1,num2;
```

وهذا يعني أنه تم الاعلان عن المتغيرين num1 و num2 على أنهما من النوع الصحيح int وهو مطابق للامر المعتاد :-

```
int num1,num2;
```

مثال 11-2) ما هو ناتج تنفيذ البرنامج التالي ؟

```
#include <iostream.h>
main()
{
	typedef int NUMBER;
	NUMBER sum,num1,num2;
	num1=10;
	num2=15;
	sum=num1+num2;
	cout << "The sum of two numbers ==>" << sum << "\n";
	return 0;
}
```

ناتج هذا البرنامج هو طباعة السطر التالي :-

```
The sum of two numbers ==> 25
```

مثال 11-3) باستخدام النوع typedef اكتب برنامجا كاملا يقوم بقراءة مصفوفتين تحتوي كل منهما على 5 قيم من النوع الصحيح، مع ايجاد حاصل ضربهما.

```
#include<iostream.h>
const MAX=5;
typedef int ARRAY[MAX];
int mult(ARRAY x,ARRAY y,ARRAY  z);
void main()
{
        ARRAY a,b,c;
        cout <<"\n*** Input data program ***\n"
             <<"\n\n Data for array 1 ==>: ";
        for(int i=0;i<MAX;i++)
        cin>>a[i];
        cout<<"\n Data for array 2 ==>: ";
        for(i=0;i<MAX;i++)
        cin>>b[i];
                            // Call function mult
        mult(a,b,c);
        cout <<"\n*** output data program***\n"
             <<"\n array1 array2 array1*array2"
             <<"\n----------------------------------";
        for(i=0;i<MAX;i++)
        cout <<"\n  \t"<< a[i] <<"  " << b[i]
             <<" \t\t"<< c[i];
}

int mult(ARRAY x,ARRAY y,ARRAY z)
{
        for(int k=0;k<MAX;k++)
        z[k]=x[k]*y[k];
}
```

تنفيذ هذا البرنامج سيعطي رسالة بادخال قيمتي المصفوفة الاولي والثانية على النحو التالي :-

```
*** Input data program ***
Data for array 1 ==>:  3 4 5 6 7
Data for array 2 ==>:  4 5 6 7 8
```

يلي ذلك اظهار نتائج ضرب المصفوفتين الذي يمثله الجدول التالي :-

```
*** output data program***
array1   array2        array1*array2
---------------------------------------------------
3        4             12
4        5             20
5        6             30
6        7             42
7        8             56
```

في هذا البرنامج تم اشهار المتغير ARRAY مصفوفة من النـوع الصحيـح طول عناصرها 5 قبل بداية الدالة الرئيسية عن طريق استخدام النوع typedef وتـم استخدام هـذا المتغير لاشـهار متغـيرات أخـري وهـي c,b,a وبالتـالي أصبحت جميعها متغيرات علـى شكل مصفوفـات صحيحـة كل واحـد منهـا يحتوي على 5 عناصر.

تمت قراءة المصفوفتين b,a وارسالهما الى الدالـة mult حيث تـم اسـتخدام المتغير ARRAY مرة اخرى لاشهار ادلة هذه الدالة z,y,x وبالتـالي الحصـول على حاصل ضرب قيم المصفوفة x في قيم المصفوفة y وتخزيـن النـاتج فـي المصفوفة z وأخيرا ترجيع قيم المصفوفة z الى المصفوفة c بالدالـة الرئيسـية وبالتالي طباعة قيم المصفوفتين وناتج ضربهما.

2.11) الاتحاد Union

هو امكانية تخزيـن ومشـاركة أكـثر مـن متغـير ذات أنـواع مختلفـة بنفس المكان والعنوان في ذاكرة الحاسب ، وهذا يمكن المبرمج مـن اسـتغلال سـعة الذاكرة الاستغلال الامثـل حيث يقـوم المـترجم بحجـز مكـان لأكـبر متغـير،

وعندما تخصص أي قيمة الى أي من هذه المتغيرات فأن القيم الموجودة سابقا تلغى عند تنفيذ البرنامج.

الشكل العام

```
union union_name
{
      type member_1;
      type member_2;
      . . .
      type member_n;
}     union_variable;
```

أو قد يأخذ الشكل التالي:

```
union union_name
{
      type member_1;
      type member_2;
      . . .
      type member_n;
};
union union_name union_variable;
```

حيث:

union_name اسم الاتحاد ويستخدم لتعريف متغيرات أخرى في البرنامج.

member وهو العضو في الاتحاد.

union_variable اسم متغير الاتحاد والذى ياخذ أكبر حجم من الذاكرة بحسب الأعضاء المكونة له.

يمكن الوصول الى أي عنصر من عناصر الاتحاد عن طريق الآتي:-

```
union_name.member;
```

أي أن اسم متغير الاتحاد يتبعه مؤثر النقطة (.) ثم العضو المعني.

مثـال 11-4) البرنـامج التـالي مهمتـه اسـناد قيمتيـن مختلفتيـن الـى عنصريـن مختلفين تحت اسم موحد.

```
#include<iostream.h>
main()
{
        union  example
        {
              int x;
              char y;
        } sample;
        sample.x=66;
        sample.y='A';
        cout <<"This value of sample.x ==>: "<<sample.x<<endl
              <<"This value of sample.y ==>: " <<sample.y<<endl;
}
```

اذا ما نفذ هذا البرنـامج سيطبع النـاتج المشابه للآتـي:-

This value of sample.x ==>:65

This value of sample.y ==>:A

هنا تم الاعلان عن المتغيرين الاول من النوع الصحيح والثاني من النوع الحرفي كعنصرين تحت اسم موحد هو sample ، والـذي يـأخذ أي قيمـة مـن أي نوع من عناصره المحددة له، حيث خصصت القيمـة 66 للعنصـر x عـن طريق جملة التخصيص

sample.x=66;

باستخدام اسم متغير الاتحاد sample يتبعـه مؤثـر النقطـة (.) يليـه العضـو وهو المتغير x وبنفس الطريقة خصصت القيمة الحرفية A للمتغير y .

نلاحظ عند تنفيذ هذا البرنامج أن العضو x له القيمة 65 غير القيمة المخصصة له اصلا وهي القيمة 66 والسبب أن كلا من العضوين x و y يأخذان نفس الحجم من الذاكرة ، وبالتالي فانهما يشاركان في نفس المكان والقيمة المدخلة أخيرا ، عليه تمت طباعة القيم العددية 65 وهي نفس القيمة الحرفية للحرف A بنظام الشفرة آسكي (ASCII) أنظر ملحق (2).

مثال 5-11) المطلوب كتابة برنامج كامل مهمته ادخال رقم الصنف في سلعة معينه بمخزن المبيعات، مع ادخال عدد المبيعات لكل صنف من الاصناف في كل شهر من الشهور الثلاثة الاولى من السنة، على ان ينتج من هذا البرنامج، طباعة البيانات المدخلة مع مجموع عدد المبيعات في الثلاثة أشهر المدخلة الى كل صنف.

```
#include<iostream.h>
const MONTH=3;
const MAX=4;
main()
{
    union target
    {
        int a[5];
        int item_number[MONTH];
    } item;
    union target sale[MAX];
    int total[MAX]={0,0,0,0};
    cout <<"\tPlease enter"
        <<"\n\t****************\n"
        <<"Item no Month1 Month2 Month3\n";
    for(int k=0;k<MAX;k++)
    {
        cin>>item.item_number[k];
        for(int i=0;i<MONTH;i++)
        cin>>sale[k].a[i];
    }
    cout <<"\n Now data with total amount of sales\n"
```

```
        <<"***********************************\n\n"
        <<"Item no  Month1  Month2 Month3   total\n"
        <<"--------------------------------\n";
    for(int i=0;i<MAX;i++)
        for(k=0;k<MONTH;k++)
            total[i]=total[i]+sale[i].a[k];
    for (k=0;k<MAX;k++)
    {
        cout << " "<< item.item_number[k];
        for(i=0;i<MONTH;i++)
        cout <<  "      " << sale[k].a[i] ;
        cout << "      " <<total[k]<<endl;
    }
}
```

عند تنفيذ هذا البرنـامج ، ينتـج عنـه رسـالة مطالبـة بادخـال رقـم الصنـف يتبعه عدد المباع منه في الشهر الاول والثاني و الثالث لعدد اربعـة اصنـاف كالآتي:-

```
Please enter
****************
Item no   Month1   Month2   Month3
1111       11       11       11
2222       22       22       22
3333       33       33       33
4444       44       44       44
```

يلي هذا الادخال رسالة اخرى توضح البيانات التـي تـم ادخالهـا مـع كميـة المبيعات لكل صنف بالشهور الثلاثة كما يلي :-

Now data with total amount of sales

****

Item no	Month1	Month2	Month3	total
1111	11	11	11	33
2222	22	22	22	66
3333	33	33	33	99
4444	44	44	44	132

الملاحظ في هذا البرنامج أنه قد تم استخدام الاتحاد union مع المتغير itemحيث يضم المتغير a من نوع المصفوفة الصحيحة وهي تمثل عدد المبيعات لكل صنف من الاصناف الاربعة، والمتغير الثاني item_number وهو يمثل عدد المبيعات في الثلاثة اشهر المعنية، وأحد أعضاء أو عناصر اسم الاتحاد المتغير itemومن ثم يصبح ممكنا ادخال رقم الصنف عن طريق الجملة التالية :-

```
cin>>item.item_number[k];
```

حيث k هنا تمثل عدد الأصناف في المخزن وهي 4 في هذا البرنامج، في حين المتغير sale فهو يأخذ شكل المصفوفة لاستعمالها في ادخال عدد الأصناف المباعة في كل شهر من الشهور الثلاثة عن طريق الجملة التالية:

```
cin>>sale[k].a[i];
```

حيث المتغير i هنا يمثل كمية الأصناف المباعة في كل شهر والتي عددها k صنف.

3.11) تراكيب البيانات Data Structures

التركيبة (Structure) تعني أمكانية تجميع مجموعة من المتغيرات تسمي بالعناصر أو الاعضاء وتكون من نفس النوع أو انواع مختلفة كل واحدة من هذه العناصر لها خانة أو حقل (Field) تحت اسم واحد بحيث يمكن الرجوع الى هذه المتغيرات عن طريق اسم التركيبة وبالتالي يمكن معالجتها كوحدة واحده ، ويتم الاعلان عن التركيبة على النحو التالي:-

```
struct struct-name
{
      type member_1;
      type member_2;
      . . .
      type member_n;
} struct-variable;
```

أو على النحو التالي:-

```
struct struct-name
{
      type member_1;
      type member_2;
      . . .
      type member_n;
};
struct struct-name struct-variable;
```

حيث:

struct الكلمة المحجوزة تعتبر هي اشهار التركيبة.

struct-name اسم التركيبة وهو اختياري.

memebr اسم العنصر أو العضو في التركيبة.

struct-variable اسم متغير له خواص التركيبة.

مثال 11-6) فيما يلي وصف لتركيبة تحت أسم BOOK

```
struct BOOK
{
     char title[25];
     char author[30];
     char publisher[25];
     float price;
     int year;
} my_book;
```

والذي يضم 5 عناصر أو اعضاء وهي كالآتي:-

1) المتغير title أى العنوان وهو من النوع الحرفي.
2) المتغير author أى المؤلف وهو من النوع الحرفي.
3) المتغير publisher أى الناشر وهو من النوع الحرفي.
4) المتغير price أى الثمن وهو من النوع الحقيقي.
5) المتغير year أى السنة وهو من النوع الصحيح.

يلي ذلك اسم متغير وهو my_book له خواص التركيبة book.

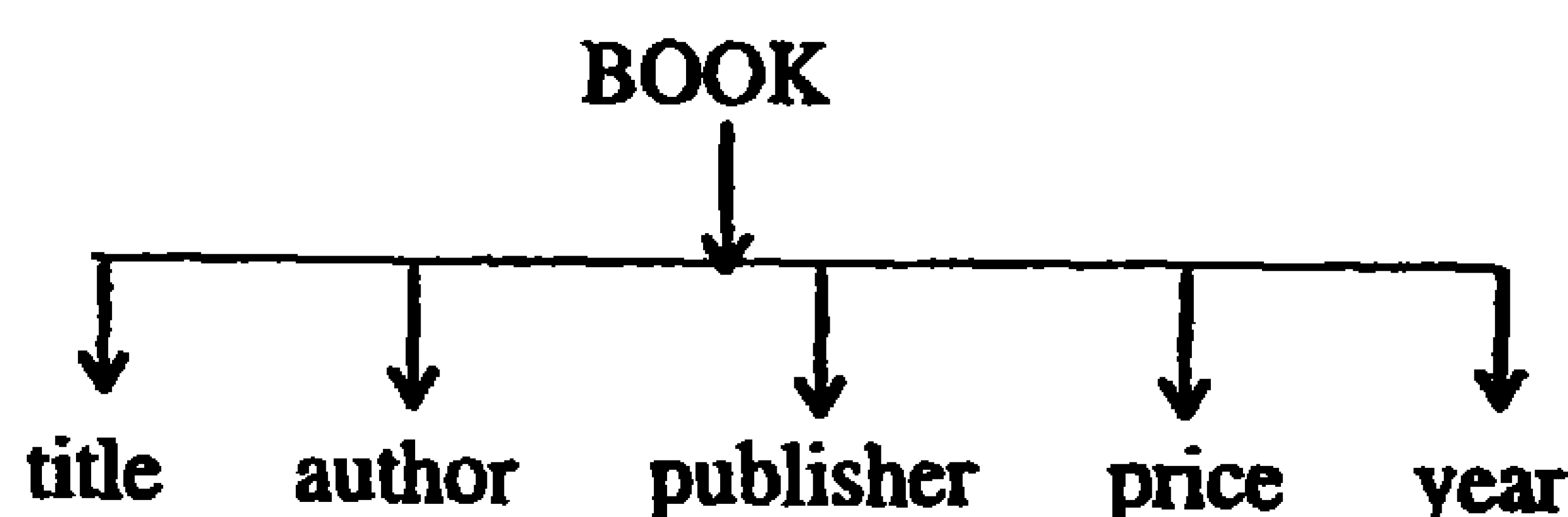
والذي نلاحظه هنا أن جميع اعضاء التركيبة محصورة بين قوسي الفئة { } وعند التعامل مع أي عضو من اعضاء التركيبة يجب التعامل معها عن طريق اسم التركيبة باستخدام الصيغة الآتية :-

```
struct-variable.member
```

أى اسم متغير التركيبة يتبعه مؤشر النقطة (.) ثم العضو المعني.

وفيما يلي التخطيط الهيكلي لهذه التركيبة :-

BOOK



مثال 7-11) البرنامج التالي مهمته طباعة تاريخ ويوم الميلاد، باستخدام مؤثر النقطة.

```
#include<iostream.h>
main()
{
        struct child
        {
              char *name;
              int day;
              int month;
              int year;
        } birthday;
        birthday.name="ALI BASHIR";
        birthday.day=12;
        birthday.month=1;
        birthday.year=1982;
        cout <<"\n My name is "<<birthday.name
              <<"\n\tand my birthday is  " << birthday.day
              <<"/"<< birthday.month<<"/"<<birthday.year;
}
```

عند تنفيذ هذا البرنامج سيظهر السطر التالي :-

```
My name is ALI BASHIR
      and my birthday is 12/1/1982
```

هنا تم الاعلان عن التركيبة تحت اسم child وهو اختياري ، وتحتوي هذه التركيبة على اربعة اعضـاء محصـورة بين قوسي الفئة يتبعها اسم متغير التركيبة birthday وهو متغير من النـوع التركيبـة child، حيث تم تخصيص قيمة صحيحة تمثل اليوم لعضو هذه التركيبة عن طريق اسم متغير التركيبة birthday يليه مؤثر النقطة (.) يتبعها اسم العضو وهو المتغير day ، وهكذا تم مع المتغير month الذي يمثل الشهر والمتغير year الذي يمثل السنة.

مثال 11-8) المطلوب تتبع هذا البرنامج وايجاد الناتج بالورقة والقلم أولا وتنفيذه على جهاز الحاسب ثانيا.

```
#include <iostream.h>
struct one
{
	char c;
	double d;
};

main()
{
	one two,three;
	two.c='&';
	two.d=100.50;
	cout<<"C="<<two.c<<"  D="<<two.d<<endl;
	three=two;
	three.d=three.d/2;
	cout<<"C="<<three.c<<"  D= "<<three.d<<endl;
}
```

4.11) التراكيب والدوال Functions and Structures

وحتي يكون البرنامج مفهوما وسهل المتابعة خصوصا مع البرامج كبيرة الحجم، وجب استخدام الدوال الفرعية كما سبق في الفصول الماضية، وحتي تكون العلاقة واضحة بين الدالة الرئيسية والدوال الفرعية التابعة لها من حيث تمرير البيانات بينها، عليه يمكن استخدام التركيبة مع هذه الدالة.

مثال 11-9) البرنامج التالي يقوم بقراءة قيم عددية من النوع الصحيح عن طريق دالة فرعية وبالتالي ارجاع وكتابة هذه القيم في الدالة الرئيسية.

```
#include<iostream.h>
int fun();
void main()
```

```
{
        int a=fun();
        int b=fun();
        cout<< "First integer ==>: "<< a<<"\n";
        cout<< "Second integer ==>: "<<b<<"\n";
}

int fun()
{
        int xy;
        cout <<"\nEnter an integer please ==>:";
        cin >>xy;
        return xy;
}
```

في هذا البرنامج تم استدعاء الدالة الفرعية والتي مهمتها قراءة قيمة صحيحة في كل مرة وارجاعها الى نقطة الاستدعاء وبالتالي طباعتها ، وفيما يلي نتائج هذا البرنامج وقت التنفيذ.

```
Enter an integer please ==>: 30
Enter an integer please ==>: 66
First integer ==>:30
Second integer ==>:66
```

مثال 11-10) يمكن أعادة كتابة البرنامج بالمثال السابق باستخدام الدالة والتركيبة.

```
#include<iostream.h>
struct two_values
{
        int x;
        int y;
}result,fun();

void  main()
{
```

```
        result =fun();
        cout<< "First integer ==>: "<< result.x<<"\n";
        cout<< "Second integer ==>: "<< result.y<<"\n";
}

struct two_values fun()
{
        struct two_values temp;
        cout <<"Enter two integer please ==>:";
        cin >> temp.x>> temp.y;
        return temp;
}
```

هنا تم الاعلان عن اسم التركيبة two_values قبل الدالة الرئيسية main() التي تضم العضوين x و y من النوع الصحيح ، واسم متغير التركيبة result والدالة fun() من نوع التركيبة two_values اما فيما يخص الدالة fun() فقد تم الاعلان عن المتغير temp من نفس نوع التركيبة المعلن عنها قبل الدالة الرئيسية وعليه فهو يضم نفس الاعضاء من حيث العدد والنوع الذين تضمهما التركيبة two_values ومن هنا نستطيع قراءة العضوين y,x عن طريق التركيبة temp ومؤثر النقطة (.) كما توضحه دالة جملة التالية :-

```
cin >> temp.x>> temp.y;
```

والرجوع بهما إلى نقطة الاستدعاء، عموما عند التنفيذ سيكون الناتج مشابها للآتي:-

```
Enter two integer please ==>: 30   66
First integer ==>:30
Second integer ==>:66
```

مثال 11-11) المطلوب اعادة كتابة البرنامج في المثال (11-7) باستخدام الدالة والتركيبة.

```
#include<iostream.h>
struct child
{
        int day;
        int month;
        int year;
        char * name;
} ;

print_date(child ddmmyy)
{
        cout <<"\n my name is "<<ddmmyy.name
              <<"\n and my birthday is"
              <<ddmmyy.day<<"/" <<ddmmyy.month
              <<"/" <<ddmmyy.year;
        return 0;
}

void main()
{
        struct child birthday;
        birthday.day=12;
        birthday.month=1;
        birthday.year=1982;
        birthday. name="ALI BASHIR ";
        print_date(birthday);
}
```

هنا تم الاشهار عن التركيبـة child خـارج نطـاق الدالـة الرئيسـية ولـم يتـم استخدام اسم متغير التركيبة الذي يلـي قوسـي الفئـة {} وفـي نفس الوقـت تـم الاعلان في الدالـة الرئيسية عـن المتغير birthday مـن نـوع التركيبـة child باستخدام الكلمة (struct) كالآتي:-

```
struct child birthday;
```

وبالتالي تم ارسال كل عناصر التركيبة الى الدالـة print_date عـن طريـق المتغير birthday اما فيما يخص الدالة فقد تم الاعلان عن المتغير ddmmyy من نفس نوع التركيبة birthday باستخدام اسم التركيبة child كالآتي:-

print_ date (child ddmmyy)

حيث قامت الدالـة باسـتقبال البيانـات مـن الدالـة الرئيسـية ومـن ثـم جـرى اخراجها كما هو آتي :-

My name is ALI BASHIR
and my birthday is 12/1/1982

مثال 11-12) البرنـامج التالي وظيفته تخصيـص قيـم ابتدائيـة الـى تركيبـة مـع استخدام دالة فرعية لطباعة هذه القيم.

```
#include<iostream.h>
struct child
{
        int day;
        int month;
        int year;
        char *y;
        char *day_name;
};

child information ={31,12,1999,"The day of" , "is Friday "};
write_date(const child &k)
{
        cout <<"\n "<<k.y<<k.day
             <<"/"<<k.month<<"/"<<k.year<<k.day_name<<endl;
}
main()
{
        write_date(information);
        return 0;
}
```

اذا ما نفذ هذا البرنامج ، سيعطي الناتج الموالي

The day of 31/12/1999 is Friday

تم استخدام اسم متغير التركيبة child وفي نفس الوقت خصصت قيم مبدئية لهذه التركيبة ، والذي يجب ان يؤخذ في الاعتبار في مثل هذه الحالة أن تكون هذه القيم المحصورة بين قوسي الفئة {} مطابقة من حيث النوع والعدد مع عناصر التركيبة.

بدأ البرنامج بتمرير جميع البيانات المخصصة لدالة فرعية باستخدام التركيبة information عن طريق دليلها k الذي هو من نفس نوع التركيبة التي مهمتها طباعة هذه البيانات.

5.11) التراكيب والمصفوفات Structures and Arrays

كثيرا ما تستخدم التراكيب في عملية معالجة البيانات عن طريق المصفوفات، حيث يمكن تعريف التركيبة كمصفوفة أحادية أو ثنائية على حد سواء.

مثال 13-11) خذ مثلا الاشهار

```
struct ARRAY
{
    int num1;
    int num2;
} matrix1,matrix2;
struct ARRAY matrix1[30];
```

يعني أن المصفوفة matrix1 تحتوي على 30 عنصرا كل واحدا من هذه العناصر عبارة عن تركيبة تضم عنصرين أو عضوين هما num1 , num2 وعليه يمكن اسناد قيمة صحيحة الى العضو num1 في الموقع الخامس من المصفوفة matrix1 على النحو التالي:-

```
matrix1[4].num1=77;
```

مثال 11-14) اذا ما تم الاعلان عن التركيبة التالية

```
struct ARRAY
{
      float mat[10][10];
      float value1;
}matrix;
```

عليه فالتعبير التالي:-

```
matrix.mat[3][4]=12.34;
```

يعني اسناد وتخزين القيمة الحقيقية 12.34 بالصف الرابع ، العمود الخامس من المصفوفة mat.

مثال 11-15) المطلوب كتابة برنامج كامل مهمته قراءة اسم الصنف وعدد المخازن وعدد الاصناف في كل مخزن مع ايجاد مجموع هذه الاصناف.

```
#include<iostream.h>
struct rtype
{
      int number_of_stores;
      char ItemName[20];
      int store[8];
};

      // ******* Function to print data *******
void print_rtype(struct rtype rec)
```

```
{
   cout << "\nYou have item name (( " << rec.ItemName<<" ))"
        <<"\nin  each  of  the  following" << rec. number_of_stores <<"
stores";
   for(int i=0;i<rec. number_of_stores;i++)
   cout << "\n store [" << i+1 << "] ==>" << rec.store[i];
}

          // ******* Main Function *******
void main()
{
      void print_rtype(struct rtype rec);
      struct rtype rec;
      cout << "\nType item name ==>:";
      cin.getline(rec. number_of_stores,80);
      cout << "\nType number of store you have ==>:";
      cin >> rec. Number _of_ stores;
      cout<<"\n-----------------------------------\n";
      int sum=0;
      for(int i=0;i<rec. number_of_stores;i++)
      {
           cout<<"Enter number of items in store " << i+1<<" ==>:" ;
           cin >> rec.store[i];
           sum+=rec.store[i];
      }
      print_rtype(rec);
      cout <<"\n\nThe total items you have ==>: "<< sum;
}
```

قبل بداية الدالة الرئيسية تم الاعلان عن التركيبة تحت اسم rtype والتي تضم ثلاث عناصر وهي المتغيرات التالية :-

المتغير number_of_stores يمثل عدد المخازن من النوع الصحيح .

المتغير ItemName يمثل اسم الصنف من النوع الحرفي.

المتغير store مصفوفة احادية وحجمها 8 عناصر من النوع الصحيح تمثل عدد الأصناف في كل مخزن.

أيضا تم الاعلان عن الدالة print_rtype والتي مهمتها استقبال وطباعة بيانات من نفس العدد والنوع الذين تتكون منهما التركيبة rtype تحت اسم rec.

اذا قمنا بتنفيذ البرنامج وادخال اسم الصنف كما يلي :-

```
Type item name ==>:coffe table
```

واتبعنا ذلك بادخال عدد المخازن التي بها الاصناف وليكن 3

```
Type number of store you have ==>:3
```

يلي ذلك ادخال عدد الاصناف في كل مخزن من المخازن الثلاثة:

```
Enter number of items in store 1 ==>:430
Enter number of items in store 2 ==>:690
Enter number of items in store 3 ==>:500
```

حيث يتم تخزين القيمة الاولي 430 في العنصر الاول من المصفوفة store والذي يمثله العداد i بالجملة for وبعد الحصول على مجموع هذه الاصناف بجميع المخازن تستدعي الدالة الفرعية print_rtype ومهمتها اخراج جميع البيانات المدخلة بالدالة الرئيسية وهي:-

```
You have item name (( coffe table ))
in each of the following 3 stores
store [1] ==>430
store [2] ==>690
store [3] ==>500
```

والرجوع الى مكان الاستدعاء لاخراج حاصل جمع الاصناف المدخلة كالآتي:

```
The total items you have ==>:1620
```

يمكن كتابة اسم العضو في اكثر من تركيبة وتحت نفس الاسم، خذ مثلا النموذج التالي:-

```
struct ONE
{
      char ch;
      int a;
      float b;
};
struct TWO
{
      char ch;
      int number;
};
struct ONE first;
struct TWO second;
```

نلاحظ أن المتغير ch هو عضو في التركيبتين ONE و TWO وعليه يمكـن الاشارة الى هذا العضو في التركيبة TWO كالآتي :-

```
second.ch
```

وكذلك يمكن الاشارة الى المتغير number أما في التركيبة first علــى النحـو التالي :-

```
first. number
```

أو في التركيبة second على النحو التالي :-

```
second. number
```

بدون أي ارباك.

خذ مثالا آخر

```
struct address
{
      char name[30];
      int street;
      char city[25];
      long phone;
};
```

هنا يمكن استخدام التركيبة تحت اسم العنوان address التي تضم (الاسم والشارع والمدينة ورقم الهاتف) في اشهار تركيبة اخرى تحت اسم الموظف employee كمايلي:-

```
struct employee
{
      struct address addrs;
      float age;
      char sex;
}worker;
```

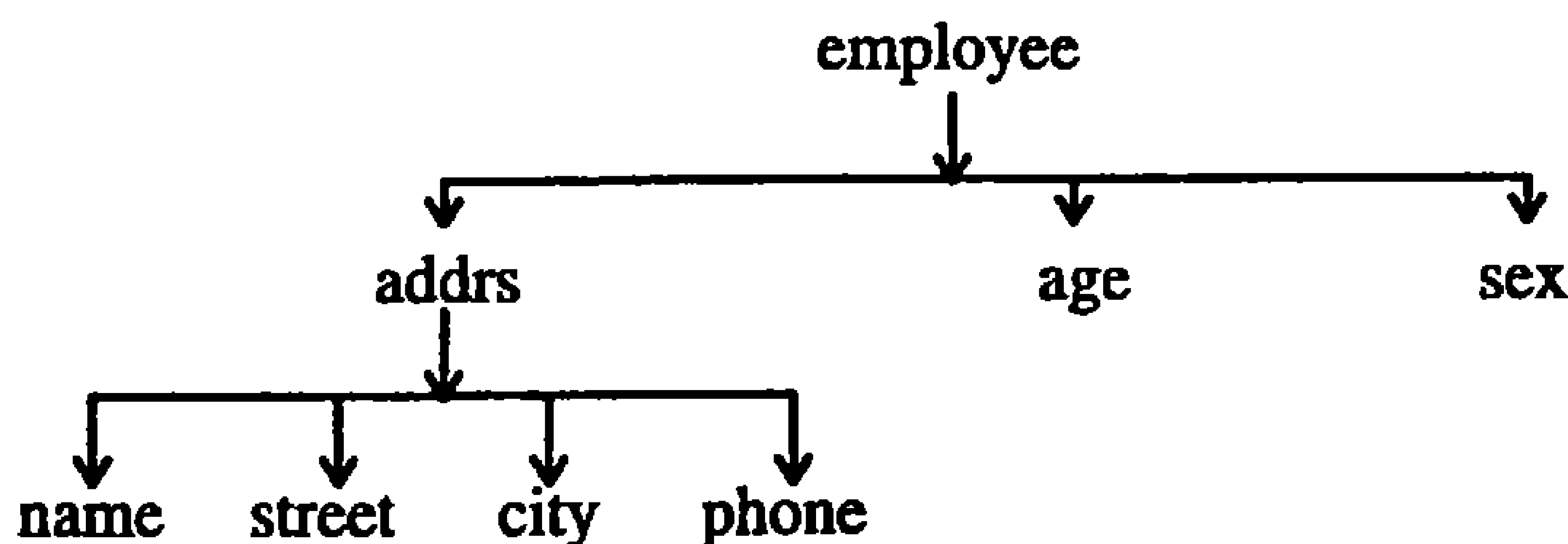
وبالتالي يمكن اسناد الاسم أو الشارع أو المدينة أو رقم الهاتف الى متغير التركيبة worker ، فعلى سبيل المثال الاسم MEERATH يتم تخصيصه للعضو name كالآتي:-

```
worker.addrs.name="MEERATH";
```

بينما اسم المدينة TAGURA TRIPOLI يمكن اسناده للعضو city كالآتي :-

```
worker.addrs.city="TAGURA TRIPOLI"
```

وفيما يلي التخطيط الهيكلي للتركيبة التي تحت اسم الموظف employee



مثال 11-16) المطلوب كتابة برنامج كامل مهمته حساب قيمة الفاتورة بعد ادخال اسم الصنف وسعره وتاريخ الشراء لعدد من الاصناف بمخزن المبيعات.

```
#include<iostream.h>
#include <conio.h>
#include<stdlib.h>
#define number 3
main()
{
	typedef struct
	{
		int day;
		int month;
		int year;
	} date;
	struct item
	{
		char product_name[20];
		char product_price[8];
		date dmy;
	};
	struct item product[number];
	float sum=0.0;
	clrscr();
	for(int i=0;i<number;i++)
	{
		cout << "Enter product name ==>";
		cin >> product[i].product_name;
		cout<<"Enter the price ==>";
		cin >> product[i].product_price;
		cout << "Enter day ==>";
		cin >> product[i].dmy.day;
		cout << "Enter month ==>";
		cin >> product[i].dmy.month;
		cout << "Enter year ==>";
		cin >> product[i].dmy.year;
	}
	for(i=0;i<number;i++)
		sum+=atof(product[i].product_price);
	cout << "--------------------------------------\n";
	cout << " no  product name  product price invoice date \n";
	cout << "--------------------------------------\n";
	for(i=0;i<number;i++)
	{
```

```
        cout <<i+1<<"    " <<"\t"<< product[i].product_name;
        cout << "\t\t" <<product[i].product_price<<"    ";
        cout << " "<<product[i].dmy.day<<"/"<<product[i].dmy.month;
        cout <<"/"<<product[i].dmy.year<<"\n";
    }
    cout << "------------------------------------\n"
        << "\n\t Total price=   " << sum;
    return 0;
}
```

تم ادماج التركيبة مـع النـوع typedef فـي الاعـلان عـن التركيبـة date أي التاريخ وتضم ثلاثـة اعضـاء وهـي اليـوم والشـهر والسنه وكلهـا مـن النـوع الصحيح ، مباشرة جاء الاعلان عن تركيبة ثانية تحت اسم item وهـي تضـم اسم الصنف والسعر اضافة الى ذلـك اعضـاء التركيبـة date تحـت اسـم dym وبالتالي فان التركيبة item اصبحت تضم خمسة اعضاء.

ايضا تم الاعلان عـن مصفوفـة product كتركيبـة مـن نفس تركيبـة item والتي مهمتها التعامل مع خمسة اعضاء لعدد number من الاصناف ، وعليـه يمكن الاشارة الى الاعضاء التي تخص الصنف الاول كما يلي:-

1) الاسم product[0].name
2) الثمن product[0].price
3) اليوم product[0].dmy.day
4) الشهر product[0].dmy.month
5) السنة product[0].dmy.year

عند تنفيذ هذا البرنامج سيتم تنظيف الشاشة ويتم ادخال البيانـات المطلوبـة بالنسبة للصنف الاول على النحو التالي :-

```
Enter product name ==> book
Enter the price ==> 190.75
Enter day ==> 12
Enter month ==> 7
Enter year ==> 1999
```

بعدها يتم تنظيف الشاشة مرة أخرى ، والمطالبة بادخال البيانـات للصنـف الثـاني، ويتكـرر هـذا الحـوار حتـي تنتهـي عمليـة ادخـال البيانـات إلـى بقيـة الاصناف، يلي ذلك جملة for التي مهمتها تنفيذ الجملة.

```
sum+=atof(product[i].product_price);
```

وفيها استعملت الدالة atof() والتي سبق شرحها ومهمتها تبديل الحرف char في آسكي (ASCII) الى عدد حقيقي، ومن ثم ايجاد مجموع اسـعار الاصنـاف المباعة، اخيرا يتم اخراج جميع البيانات المدخلة مع قيمة الفاتورة، وقد تكـون مشابهة للآتي:-

```
--------------------------------------------------
no  product name  product price invoice date
--------------------------------------------------
1       book          190.75        12/7/1999
2       pen           66.50         21/7/1999
3       pencil        75.25         25/7/1999
--------------------------------------------------

        Total price = 332.50
```

6.11) التركيبات والمؤشرات Structures and Pointers

للتعامل مع اعضاء أي تركيبة معقدة باستخدام المؤشرات هنـاك طريقـة أخرى وهي استعمال مؤثر التركيبة (<-) أي مؤثر الطـرح (-) يتبعهـا مؤثـر أكبر من (>) كما يوضحه الشكل التالي:-

```
strcut_variable->member;
```

حيث strcut_variable متغير اسم التركيبة يتبعه المؤشر (<-) ثم عضو التركيبة member مع ملاحظة أن مؤثر النقطة (.) ومؤثر التركيبة (<-) لهما الأسبقية القصوى من بين مؤثرات لغة سى++

مثال 11-17) الجملة

```
q->sum=0;
```

تعني تخزين القيمة 0 في المتغير sum المشار اليه بالمؤشر q.

مثال 11-18) اعادة كتابة البرنامج في المثال (11-15) الذي يحسب مجموع عدد الاصناف في عدد من المخازن باستخدام مؤثر التركيبة (<-).

```
#include<iostream.h>
struct rtype
{
        int ItemNum;
        char ItemName[20];
        int store[8];
};

            // ******* Main function *******
void main()
{
        int Struct_Pointer(struct rtype *rec);
        struct rtype rec;
        int sum=Struct_Pointer(&rec);
        cout<<"\n\nThe total items you have ==>: "<< sum;
        getch();
}

        // ******* Function to input and output data *******
int Struct_Pointer(struct rtype *rec)
{
        int sum=0;
        cout << "\nType item name ==>:";
        cin.getline(rec->ItemName,80);
        cout << "\nType number of store you have ==>:";
```

```
    cin >> rec->ItemNum;
    cout<<"\n--------------------------------\n";
    for(int i=0;i<rec->ItemNum;i++)
    {
        cout<<"Enter number of items in store " << i+1<<" ==>:" ;
        cin >> rec->store[i];
        sum+=rec->store[i];
    }
    cout << "\nYou have item name (( " << rec->ItemName<<" ))"
        << "\nin each of the following " << rec->ItemNum<<" stores";
    for( i=0;i<rec->ItemNum;i++)
        cout << "\n store [" << i+1 << "] ==>" << rec->store[i];
    return (sum);
}
```

في هذا البرنامج عندما استدعيت الدالة الفرعية Struct_Pointer تم استخدام المتغير rec من النوع المؤشر وذلك بأن سبقه المؤشر (*) وايضا فهو يعتبر من نوع التركيبة rtype، وعليه يتم التعامل مع هذا المؤشر باستخدام مؤثر التركيبة (<-) في هذه الحالة.

بدأت هذه الدالة بالجملة

```
cin.getline(rec->ItemName,80);
```

ومهمتها استقبال اسم الصنف من لوحة المفاتيح وتخزينه في المتغير الحرفي ItemName المشار اليه بالمؤشر rec ويمكن تخصيص الاسم coffe table لهذا المتغير كالآتي :-

```
rec->ItemName="coffe table";
```

اما جملة الادخال الثانية

```
cin >> rec->ItemNum;
```

فهي تختص باستقبال قيمة عددية وليكن الرقم 3 مثلا ليمثل عدد المخازن وتخزينها في المتغير الصحيح ItemNum المشار اليه بالمؤشر rec وهي تكافئ جملة التخصيص التالية :-

```
rec->ItemNum=3;
```

اخيرا تم استخدام جملة for ومهمتها استقبال عدد الاصناف في كل مخزن عن طريق الجملة

```
cin >> rec->store[i];
```

وتخزينها في المصفوفة store وفي نفس الوقت جمع قيم عناصر هذه المصفوفة وحفظها بالمتغير sum وطباعة عدد الاصناف في كل مخزن ومن ثم الرجوع الى نقطة الاستدعاء وطباعة مجموع الاصناف في كل المخازن.

ويمكن تخزين القيمتين 690,430 بالمصفوفة store باستخدام جملة التخصيص على النحو التالي :-

rec->store[1] =430; أي تخزين القيمة 430 في عنصر المصفوفة الاول المشار اليه بالمؤشر rec

rec->store[2]=690; أي تخزين القيمة 690 في عنصر المصفوفة الثاني المشار اليه بالمؤشر rec

يمكن اعادة الدالة Struct_Pointer بالبرنامج السابق كالآتي:

```
     // ******* Function to input and output data *******
int Struct_Pointer(struct rtype *rec)
{
       int sum=0;
       cout << "\nType item name ==>:";
       cin.getline((*rec).ItemName,80);
       cout << "\nType number of store you have ==>:";
```

```
    cin >> (*rec).ItemNum;
    cout<<"\n--------------------------------\n";
    for(int i=0;i<(*rec).ItemNum;i++)
    {
        cout<<"Enter number of items in store " << i+1<<" ==>:" ;
        cin >> (*rec).store[i];
        sum+=(*rec).store[i];
    }
    cout << "\nYou have item name (( " << (*rec).ItemName<<" ))"
        << "\nin each of the following " << (*rec).ItemNum<<" stores";
    for( i=0;i<(*rec).ItemNum;i++)
        cout << "\n store [" << i+1 << "] ==>" << (*rec).store[i];
    return (sum);
}
```

في هذه الدالة الفرعية تم الاعلان عن المتغير rec من نـوع المؤشـر وهـو يؤشر الى تركيبة تضم نفس اعضاء التركيبة rtype بالدالـة الرئيسـية main() كما استخدم المؤثر (*) ومؤثر النقطة (.) عوضا عن مؤثر التركيبة (<-) ، مع ملاحظة استخدام الاقواس مع المؤثرين وهي ضرورية لان مؤثـر النقطـة (.) له الأسبقية في التنفيذ على مؤثر (*) وبالتالي يمكن اسناد القيم المعطاة بالمثال الاخير عند استخدام هذين المؤثرين كما توضحه الجمل التالية :-

```
(*rec).ItemName="coffe table";
(*rec).ItemNum=3;
(*rec).store[1]=430;
(*rec).store[2]=690;
```

اذا مـا نفذ البرنـامج باستخدام أي مـن الدالتيـن ، فسـوف يعطـي النتـائج المشابهة بالمثال (11-15).

مثال 19-11) المطلوب كتابة برنامج كامل مهمته الاولى قراءة بيانات تخص فصلا به عدد معين من الطلبة على ان يتم ادخال هذه البيانات لكل طالب على هيئة تركيبة تضم العناصر التالية:-

1) المتغير idno من النوع الصحيح ويمثل رقم الطالب.
2) المتغير exam عبارة عن مصفوفة احادية من النوع الحقيقي وتضم درجات امتحانات لعدد 3 مقررات لكل طالب.

ومهمته الثانية طباعة كل البيانات المدخلة مع حساب ما يلي :-

1) متوسط درجات المقررات الثلاثة لكل طالب .
2) حساب متوسط درجات كل مقرر بالفصل
3) حساب اعلى درجة في المقرر الاول
4) حساب اعلى درجة في المقرر الثاني
5) حساب اعلى درجة في المقرر الثالث

```
#include <iostream.h>
#include <string.h>
#define num 2     // ******* number of students  *******
#define tests 3   // *******   number of tests   *******
float max[tests],
avg[tests],total_exam[tests];
struct student
{
        int idno;
        float exam[tests];
};

        // ******* Function computes total grade *******
        //              for each student
int  total (std,total_exam)
struct student *std[num];
float total_exam[num];
{
        for(int i=1;i<=num;i++)
```

```
        {
             float sum=0.0;
             for(int j=1;j<=tests;j++)
                    sum+=std[i]->exam[j];
             total_exam[i]=sum;
        }
}

        // ******* Function computes the average *******
        //            of each test
int the_avg(std,avg)
struct student *std[num];
float avg[num];
{
        for(int j=1;j<=tests;j++)
        {
             float sum=0.0;
             for(int i=1;i<=num;i++)
                    sum+=std[i]->exam[j];
             avg[j]=sum/num;
        }
}

        // *******  Function computes the maximum *******
        //            grade for each test
float the_maxs(std,max)
struct student *std[num];
float max[tests];
{
        for(int i=1;i<=tests;i++)
        {
             max[i]=std[0]->exam[i];
             for(int j=1;j<num;j++)
                    if(std[j]->exam[i]>max[i])
                           max[i]=std[j]->exam[i];
        }
        return 0;
}
             //  *******  main function  *******
void main()
{
```

```
      struct student *std[num];
      for(int i=1;i<=num;i++)
      {
           cout << "Enter idno[" << i << "] ==>:" ;
           cin >> std[i]->idno;
           for(int j=1;j<=tests;j++)
           {
                  cout << "Enter exam[" << j << "] ==>:";
                  cin >> std[i]->exam[j];
           }
           cout << "\n";
      }
      total(std,total_exam);
      the_avg(std,avg);
      the_maxs(std,max);
      cout << "-----------------------------------------\n"
           << " id no    test1  test2  test3  average\n"
           << "-----------------------------------------\n";
      for(i=1;i<=num;i++)
      {
           cout << std[i]->idno <<"    ";
           for(int j=1;j<=tests;j++)
                  cout <<"\t" <<std[i]->exam[j];
           cout << "\t"<< total_exam[i]/tests<<"\n";
      }
      cout << "\n-----------------------------------------\n";
      for(i=1;i<=tests;i++)
           cout << "\nThe average of test " << i << " ==>: "<< avg[i];
      cout << "\n\n";
      for(i=1;i<=tests;i++)
         cout << "\nThe maxamum grade in test "<<i<< " ==>: "<<max[i];
      getch();
}
```

اذا ما نفذ هذا البرنامج سيترتب عليه:

1) ادخال البيانات وهي رقم القيد للطالب الاول مع درجات امتحان المقررات الثلاثة وقد تكون كالآتي :-

```
Enter idno[1] ==>: 990501
Enter exam[1] ==>: 87
Enter exam[2] ==>: 81
Enter exam[3] ==>: 90
```

يتبعها ادخال البيانات التي تخص الطالب الثاني وهي:

```
Enter idno[2] ==>: 990577
Enter exam[1] ==>: 75
Enter exam[2] ==>: 83
Enter exam[3] ==>: 69
```

2) اخراج البيانات المدخلة لكل طالب مع متوسـط درجاتـه فـي كـل مقـرر وهو ما يوضحه الجدول التالي :-

```
---------------------------------------------------------
id no      test1   test2   test3   average
---------------------------------------------------------
990501     87      81      90      86
990577     75      83      69      75.666664
---------------------------------------------------------
```

اخيرا مجموعة من المعلومـات التـي تخـص كـل طـالب بالنسبة لجميـع المقررات وهي كما يلي:-

```
The average of test 1 ==>:81
The average of test 2 ==>:82
The average of test 3 ==>:79.5

The maxamum grade in test 1 ==>:87
The maxamum grade in test 2 ==>:83
The maxamum grade in test 3 ==>:90
```

7.11) تمارين Exercises

(1) اذكر الفرق بين كل من

(a) Typedef (b) Union (c) Structure

مع اعطاء بعض الأمثلة لكل واحدة منها.

(2) كيف تتم الاشارة الى أي عضو داخل تركيبة معينة؟ اعط مثلا على ذلك.

(3) المطلوب تعريف تركيبة تحتوي العناصر أو الاعضاء التالية:-

الاول من النوع الحرفي تحت اسم name

الثاني من النوع المضاعف number

الثالث من نوع مصفوفة صحيحة int طولها 20

(4) اكتب برنامجا لتصميم تركيبة تخص فاتورة استهلاك الكهرباء تضم اسم المستهلك، رقم العداد، قيمة الاستهلاك، وتركيبة داخلية تخص نوع الاستهلاك وتضم (مساكن ، ورش ، مزارع ، محلات تجارية)، ثم اطبع التالي:-

* فاتورة تضم كل البيانات الخاصة بالمزارعين مع المجموع الكلي لقيمة استهلاكهم للكهرباء.
* كل البيانات التي تخص اعلى مستهلك خاص بالمساكن.
* قائمة تضم كل البيانات السابقة للمستهلكين (ورش ومحلات تجارية).

(5) اكتب التركيبة اللازمة تحت اسم الدواء (drug) تضم الأعضاء الآتية:-

* اسم الدواء name من النوع الحرفي.
* رقم جهة انتاج الدواء id comp من النوع الصحيح.
* العنوان address الذى يضم (الشارع street من النوع الحرفي والمدينة city من النوع الحرفي).

* تاريخ انتهاء الصلاحية expire date ويضم الأعضاء (اليوم والشهر والسنة)
* العدد count من النوع الصحيح.
* السعر price من النوع الحقيقي.

(6) صف ناتج البرامج التالية :-

(a)

```
#include<iostream.h>
main()
{
	union example
	{
		char *x;
		char *y;
		char *z;
	} sample;
	sample.x="GREEN";
	cout <<"X ==>"<<sample.x<<"  Y ==>" <<sample.y
		<<"  Z ==>"<<sample.z<<endl;
	sample.y="BLUE";
	cout <<"X ==>"<<sample.x<<"  Y ==>" <<sample.y
		<<"  Z ==>"<<sample.z<<endl;
}
```

(b)

```
#include<iostream.h>
struct numbers
{
	int i;
	float f;
	double d;
} ;
print_data(numbers sample)
{
	cout <<"A ==>:"<<sample.i<<"  B ==>:"<<sample.f
		<<"  C ==>:"<<sample.d;
```

```
}

void main()
{
        struct numbers ss={10,5.6,0.0012};
        print_data(ss);
}
```

(c)

```
#include<iostream.h>
struct numbers
{
        int i;
        float f;
        double d;
} ;

print_data(numbers *sample)
{
        sample->i=11;
        sample->f=2.2;
        sample->d=123.005;
}

void main()
{
        struct numbers ss={10,5.6,0.0012};
        cout <<"Data before call :"<<endl;
        cout <<"A ==>:"<<ss.i<<"  B ==>:"<<ss.f
             <<"  C ==>:"<<ss.d<<endl;
        cout <<"Data after call :"<<endl;
        print_data(&ss);
        cout <<"A ==>:"<<ss.i<<"  B ==>:"<<ss.f
             <<"  C ==>:"<<ss.d;
}
```

(7) صمم تركيبة لوصف عدد من الفنادق، حيث يتم قراءة الاسم ، العنوان، تكلفة المبيت ،انواع الوجبات. ثم اكتب:

* دالة مهمتها طباعة هذه الفنادق حسب التكلفة في ترتيب تنازلي من اعلى تكلفة إلى أقل تكلفة.
* ودالة اخرى مهمتها طباعة اسم الفندق الذي له اعلى تكلفة.
* دالة وظيفتها طباعة قائمة بالوجبات التي تقدم.

(8) اكتب برنامجا به التركيبة child_data التي تضم اسم المولود مكان الولادة، الجنس، التاريخ الذي يضم العناصر (اليوم والشهروالسنة) ثم باستخدام الدالة:-

* الاولى اطبع اسماء الذكور مع العناوين بترتيب تصاعدي حسب الاسماء.
* الثانية اطبع قائمة تضم اسماء الاناث وتاريخ ولادتهن مع العناوين.

(9) اكتب برنامجا كاملا يقرأ عدد N من السائقين ، باستخدام typedef صمم تركيبة تخص مركز الشرطة تضم العناصر اسم السائق، رقم الرخصة، عمر السائق، الجنس، نوع الرخصة (أولي ، ثانية ، ثالثة)، تاريخ الاصدار.

ثم اكتب عددا من الدوال مهمتها عمل الآتي :-

* طباعة كل البيانات التي تخص اصغر سائق أولاً واكبر سائق ثانياً.
* طباعة تقرير يضم ارقام الرخص مع اسماء كل السائقين الحاملين للرخصة من الدرجة الثانية والذين تقل اعمارهم عن 40 سنة.
* طباعة تقرير آخر به كل اسماء الذكور والحاملين للرخص من الدرجة الثالثة مع تواريخ الاصدار.

الفصل الثاني عشر: معالجة الملفات

في البرامج التي كتبناها في الفصول السابقة كنا نقوم بادخال البيانات عن طريق لوحة المفاتيح ونعالج هذه البيانات ثم تخزنها في ملفات مؤقته (Temporary Files) في ذاكرة الحاسب أو نخرجها على الشاشة (Screen) الا أن هذه الطريقة بها عيب واضح جدا وهو أن هذه الملفات سرعان ما تفقد البيانات المدونة بداخلها بمجرد الانتهاء من تنفيذ البرنامج الأمر الذي يفرض على المستخدم اعطاء البيانات المناسبة للبرنامج كلما رغب في تكرار تنفيذه، وهذه عملية متعبة وخصوصا اذا كانت البيانات كبيرة الحجم.

عليه فقد بات من الضروري الاستعاضة عن هذه الملفات المؤقتة بالملفات الدائمة التخزين (Permanent Files) ، حيث تمتاز هذه الملفات بصلاحيتها لحفظ قدر كبير من البيانات وامكانية الرجوع اليها وقت الحاجة .

1.12) تعريف وانشاء الملف Defining and Creating a File

قبل الدخول في معالجة الملفات (File Processing) ، يجب تعريف الملف، فالملف (File) عبارة عن تركيبة بيانية تضم العديد من السجلات(records) حيث يتكون كل سجل من مجموعة من الخانات أو الحقول (fields)، وكل حقل يتكون من مجموعة من الحروف أو الارقام أو الرموز أو جميعها، مثل الاسم والعنوان ورقم الهاتف لأى موظف، والملفات تنقسم الى فئتين من حيث

طريقة التخزين الدائم للملفات النصية Text Files والمفات الثنائية Binary Files وسوف نتطرق إلى شرح الفرق بينهما فيما بعد إن شاء الله تعالى.

قبل التعامل مع الملف ، وجب اتباع الآتي:-

1) وجب استخدام ملف العناوين fstream.h الذي عن طريقـه يتم الاتصـال بين البرنـامج والملف.

2) فتح الملف أو ربطه يتم بالاعلان عنه بواسطة احدى الطرق التالية:-

الادخال	ifstream
الاخراج	ofstream
الادخال والاخراج	fstream

باستخدام الدالة open لاقامة الاتصال بين البرنامج والملف كالآتي :-

```
FStreamName.open(FileName,Mode);
```

حيث FStreamName اسم القناة أو الاسلوب وذلك لربط الملف FileName من المكان الذي تتم فيه قراءة أو كتابة البيانات باستخدام Mode وهي نمط أو حالة عمل فتح الملف ، والجدول التالي يبين كيفية التعامل مع Mode

mode	المعنى
ios::app	فتح الملف للاضافة في نهايتة.
ios::ate	ضع المؤشر على نهاية الملف عند فتحه.
ios::in	فتح الملف للكتابة.
ios::out	فتح الملف للقراءة.
ios::trunc	امسح الملف اذا كان موجودا.
ios::nocreate	فشل عملية فتح الملف اذا كان غير موجود.
ios::noreplace	نجاح عملية فتح الملف اذا كان موجودا.

3) اغلاق الملف الذي تم فتحه عن طريق الدالة (close) بالأمر:-

```
FStreamName.close();
```

2.12) الملفات النصية Text Files

الملفات النصية هي عبارة عن أسطر متعددة تحتوي بيانات من نوع رموز الشفرة آسكي (ASCII) القابلة للطباعة وفي كل سطر له نهاية محددة لتمييز كل سطر من اسطر النص عن بقية السطور ، وهذا النوع من الملفات يمكن قراءته والتعرف عليه من قبل الانسان.

1) الكتابة في ملف Writing in a File

حتي يمكن كتابة أية بيانات في ملف معين ، وجب استخدام بعض الجمل مثل فتح الملف والكتابة عليه واخيرا قفل الملف.

مثال 12-1) البرنامج التالي مهمته تكوين ملف مع قراءة رقم الموظف واسمه وعدد ساعات الشغل الاضافية في الاسبوع وحفظ هذه البيانات في الملف لاستعماله لاحقا.

```
#include<conio.h>
#include<iostream.h>
#include<fstream.h>
main()
{
        ofstream Data_file;    // Declare the file
        long num;
        char name[50];
        float hours;
        clrscr();
                              // Open a file overtime.dat
        Data_file.open("overtime.dat ",ios::out);
```

```
    cout<<"Enter number, hours and name of 3 employees :";
    for(int i=0;i<3;i++)
    {
        cout<<"\nEnter employee number ==>:";
        cin>>num;
        cout<<"Enter number of hours ==>: ";
        cin>>hours;
        cout<<"Enter employee name ==>: ";
        cin>>name;
                        // write data in the file
        Data_file<<num<<"    "<<hours<<"    "<<name<<endl;
    }
    Data_file.close();    // close file overtime.dat
    return 0;
}
```

لقد تم الاشهار عن Data_file لفتح قناة الملف للاخراج باستخدام الفصيلة ofstream كما هي معرفة في ملف العناوين fstream.h كما يلي:

```
ofstream Data_file;
```

اما الجملة

```
Data_file.open("overtime.dat ",ios::out);
```

فهي تعني فتح ملف تحت الاسم overtime.dat وربطه بالقناة المفتوحة للاخراج Data_file وقد استخدمت الصيغة (ios::out) وهذا يعني أن الملف قد اعد للكتابة فيه ، وان وجد هذا الملف سابقا فسوف يتم الغاء البيانات منه.

مهمة هذا البرنامج هو ادخال عدد ثلاثة سجلات لعدد ثلاثة موظفين عن طريق جملة for بحيث يضم كل سجل ثلاثة حقول هي رقم الموظف num واسمه name وعدد ساعات الشغل الاضافية في الاسبوع hours عن طريق دالة قناة الادخال ، يأتي بعدها كتابة هذه السجلات على الملف Data_file باستخدام مؤثر >> مع اسم القناة التي تم ربطها بالملف على القرص بالفهرس الحالي عن طريق الجملة:

```
Data_file<<num<<"     "<<hours<<"    "<<name<<endl;
```

بعد نهاية جملة for ، يتم اغلاق الملف في نهاية البرنامج باستخدام الامر

```
Data_file.close();
```

عموما وقت تنفيذ البرنامج السابق ستظهر الرسالة الاتية :-

```
Enter number, hours and name of 3 employees :
```

طالبة ادخال رقم واسم والساعات لعدد 3 موظفين وقد تكون كالآتي :-

```
Enter employee number ==>:9901
Enter number of hours ==>:15
Enter employee name ==>: Sami
Enter employee number ==>:9910
Enter number of hours ==>:20
Enter employee name ==>: Aasum
Enter employee number ==>:9925
Enter number of hours ==>:25
Enter employee name ==>: Kadija
```

سيتم حفظ هذه البيانات في الملف overtime.dat على هيئة ثلاثة سجلات.

وللتأكد من ان البرنامج قد أنشأ ملفا جديدا تحت اسم overtime.dat، يمكن تنفيذ أمر التشغيل (DIR) من النظام (DOS)، أيضا لفحص محتويات هذا الملف يمكن استعمال الامر (TYPE) بالشكل التالي:-

```
TYPE overtime.dat
```

وسوف تشاهد على شاشة العرض ثلاثة سجلات وهي التي يحتوي عليها هذا الملف.

2) الإلحاق الى ملف Appending to File

في بعض الأحيان يتحتم على المبرمج إلحـاق (Appending) بيانـات قد تـم نسيانها أو استحداث بيانات جديدة وإلحاقهـا الـى ملـف سبق انشـاؤه ، المثـال التالي يبين طريقة الاستخدام.

مثال 12-2) على فرض اننا نريد اضافة بيانـات تخص عددا مـن الموظفيـن الآخرين الى السجلات الموجودة بالملف القديم overtime.dat الـذي تم انشـاؤه في المثال (12-1) ، عليه ينبغي تنفيذ البرنامج السابق مع تغيير دالة فتح الملف فقط

```
Data_file.open("overtime.dat ",ios::out);
```

لتصبح

```
Data_file.open("overtime.dat ",ios::ate);
```

حيث جرى استخدم الصيغة (ios::ate) للدلالة على اضافة وكتابة البيانـات فـي نهايـة الملـف overtime.dat، واذا مـا ادخلـت البيانـات التاليـة لعـدد 2 مـن الموظفين:

```
Enter employee number ==>:9926
Enter number of hours ==>:30
Enter employee name ==>: Enas
Enter employee number ==>:9935
Enter number of hours ==>:20
Enter employee name ==>:Hajer
```

سيحتوي الملف الآن على بيانات لعدد 5 موظفين.

3) قراءة الملف Reading The File

كما تم فتح ملف ووضعت فيه عدد من البيانات، يمكن فتحه مرة اخرى وبالتالي القراءة منه للتأكد من صحة البيانات المحفوظة فيه، وحتى يتم شرح كيفية القراءة نورد المثال التالي:-

مثال 12-3) البرنامج الموالي مهمته قراءة البيانات الموجودة في الملف overtime.dat ومن ثم حساب قيمة الشغل الاضافي الاسبوعي على اساس 10 دينارات للساعة الواحدة لكل موظف وعرضها على الشاشة.

```
#include<conio.h>
#include<iostream.h>
#include<fstream.h>
#include<stdlib.h>
main()
{*
        ifstream Data_file;
        long id;
        char sname[50];
        float hours;
        clrscr();
                            // Open file overtime.dat
        Data_file.open("overtime.dat ",ios::in);
        if(!Data_file)
        {
            cerr<<"Error file not found";
            getch();
            exit(1);
        }
        cout<<"\n Employee number   employee name   hours   amount\n";
        cout<<"----------------------------------------------\n";
                            //  read number, name and hours
        for(int i=0;i<5;i++)
        {
            Data_file>>id>>hours>>sname;
            cout <<id<<"\t"<<sname<<"\t"<<hours
```

```
            <<"\t"<<hours*10<<endl;
    }
            //  close the file
    Data_file.close();
    getch();
}
```

بعد اشهار وفتح الملف عن طريق الجملة التالية :-

```
Data_file.open("overtime.dat ",ios::in);
```

والتي تعني أن اسم الملف overtime.dat قد أعد للقراءة منه فقط حيث تم استخدام الصيغة (ios::in) في هذه الحالة.

يمكن الاعلان عن نوع الفصيلة أو القناة وفتح الملف في جملة واحدة وبدون استخدام دالة open كالآتي :

```
ifstream Data_file("overtime.dat ",ios::in);
```

أما جملة if التي بالشكل

```
if( !Data_file)
```

أو بالشكل

```
if(Data_file.fail())
```

فكلاهما تعني أنه اذا كانت عملية فتح الملف overtime.dat فاشلة أي انه غير موجود، يعطي البرنامج باستخدام الهدف (cerr) رسالة الخطأ التالية:-

Error file not found

وبالتالي الخروج نهائيا من البرنامج عن طريق دالة الخروج exit(1) وهذا ما يجب أن يتبع قبل استخدام أي ملف، وحيث ان الملف overtime.dat قد تم

انشاؤه عن طريق البرنامج في المثال (12-1) وأيضا ألحق به عدد من البيانات الجديدة بالمثال (12-2)، فيتم قراءة كل البيانات المخزنة بالملف باستخدام جملة for وبالتالي حساب قيمة المبلغ المستحق لكل موظف مع طباعة البيانات التي تخصه واخيرا غلق الملف ونهاية البرنامج، وفيما يلي النتائج في حالة تنفيذ هذا البرنامج

```
Employee  number   employee name  hours   amount
-----------------------------------------------------------
           9901           Sami      15      150
           9910           Aasum     20      200
           9925           Kadija    25      250
           9926           Enas      30      300
           9935           Hajer     20      200
```

مثال 12-4) اكتب برنامجا كاملا لقراءة مجموعة من القيم من النوع الصحيح، مع حفظ هذه القيم في ملف وايقاف عملية القراءة عندما يتم ادخال قيمة غير صحيحة (حرف أو رمز)، ثم قراءة وطباعة هذه القيم المحفوظة وطباعة حاصل جمعها.

```
#include<conio.h>
#include<iostream.h>
#include<fstream.h>
main()
{
        fstream InOutStream;
        int num, sum=0;
        char file_name[20];
        clrscr();
        cout<<"Enter file name please ==>: ";
                // print your file name
        cin>>file_name;
            // open file name to write sequnce numbers
```

```
    InOutStream.open(file_name,ios::out);
    cout<<"\nInput list of integer numbers ";
    cout<<"\nOR type any character to stop\n";
    while( cin>>num )
    InOutStream<<num<<" ";
                  //  close the file
    InOutStream.close();
              // open file to read numbers
    InOutStream.open(file_name,ios::in);
    cout<<"Data in file looks like \n";
    while(InOutStream>>num , !InOutStream.eof())
    {
        cout<<num<<" ";
        sum+=num;
    }
    cout<<"\nThe sum of all numbers ==>: "<<sum;
    getch();
                  // close the file
    InOutStream.close();
}
```

عند تنفيذ هذا البرنامج تظهر الرسالة التالية على شاشة العرض

Enter file name please ==>:

في انتظار طباعة اسم الملف المراد حفظ البيانات فيه ، وهذا يعطي المبرمج امكانية اختيار اسم الملف المناسب بدلا من الاسم الثابت وبالتالي اسناد هذا الاسم للمتغير الحرفي file_name وعند ادخال الاسم وليكن SUM.DAT والذي ينتهي بالامتداد DAT للدلالة على ان هذا الملف هو ملف بيانات، يلي ذلك اظهار الرسالة التالية :-

Input list of integer numbers

OR type any character to stop

والتي تبين كيفية ادخال البيانات وطريقة ايقاف تنفيذ البرنامج.

بالنسبة لجملة while الاولي

```
while( cin>>num )
```

فهي تعني قراءة الرقم المدخل من النوع الصحيح وكتابته في الملف باستخدام مؤشر >> مع اسم القناة التي تم ربطها كالآتي :-

```
InOutStream<<num<<" ";
```

وتتكرر عملية القراءة والتخزين حتي يتم ادخال قيمة غير عددية ، عندها يغلق الملف الذي تم فتحه ، وحيث ان الملف قد اغلق ، عليه يجب فتحه مرة اخرى لغرض القراءة منه وقد استخدمت جملة while الثانية

```
while(InOutStream>>num , !InOutStream.eof())
```

عوضا عن جملة for والسبب هو أننا قد لا نعرف عدد السجلات أو السطور بالملف مسبقا ، وتعني بينما لم يؤشر مؤشر الملف للنهاية نفذ الجمل التالية لجملة التكرار أي قراءة البيانات المحفوظة بالملف المذكور وطباعتها وايجاد مجموعها، ويتكرر هذا حتي الوصول الى نهاية الملف ، حينها يطبع المجموع وينتهي البرنامج.

وهذا هو ناتج تنفيذ البرنامج عند ادخال البيانات المناسبة

```
Enter file name please ==>: SUM.DAT

Input list of integer numbers
OR type any character to stop
6 8 2 -5 4F
```

والاخراج

```
Data in file looks like
6 8 2 -5 4
```

The sum of all numbers ==>: 15

قد يتم كتابة برنامج معين لعمل وظيفة ما ، وبالتالي فان المـبرمج مطـالب باعطائه البيانات المناسبة والتي قد تكون كثيرة جدا في كل مرة يتم فيها تنفيذ هذا البرنامج وهذه طريقة جد متعبة ومضيعة للوقت ، مما يوجب تخزين هذه البيانات مهما كان حجمها في ملف تحت اي اسم مقبول ، وبالتالي قراءة هـذه البيانات من ذلك الملف ومن ثم معالجتها وعرض النتائج على شاشة العـرض أو حفظها في ملف جديد آخر بقصد استعمالها عند الحاجة.

مثال 12-5) البرنامج التالي يوضح أسلوب قراءة بيانات من ملف نصي سبق انشاؤه تحت اسـم input.dat حيث يحتوى على مجموعة من الاسطر غير معروف عددها ، ويقوم بايجاد الاحصائية التالية :-

1) عدد الفراغات.
2) عدد الحروف الهجائية.
3) عدد الأرقام.
4) عدد الأسطر.

مع حفظ هذه الاحصائية في ملف آخر تحت اسم output.dat

```
#include<ctype.h>
#include<conio.h>
#include<iostream.h>
#include<fstream.h>
#include<stdlib.h>
main()
{
        char ch;
        char in_data[20], char out_data[20];
        fstream in_file ;fstream out_file;
        clrscr();
```

```
cout<<"Enter data file name ==>: ";
cin>>in_data;
cout<<"Enter output data file ==>: ";
cin>>out_data;
                // open input file
in_file.open(in_data,ios::in);
if(!in_file)
{
    cerr<<"File ["<<in_data<<"] could not be opened";
    getch();
    exit(1);
}
                // open output file
out_file.open(out_data,ios::out);
if(!out_file)
{
    cout<<"File ["<<out_data<<"] could not be opened";
    getch();
    exit(1);
}
int blank=0,letter=0,digit=0,line=0;
while(in_file.get(ch) , !in_file.eof())
{
    if(ch=='\n')line+=1;
    if(ch==' ')blank+=1;
    if((ch>='a' && ch<='z')||(ch>='A'&& ch<='z'))letter+=1;
    if(ch>='0' && ch<='9')digit+=1;
}
                    // close input file
in_file.close();
    // Write the all information in output file
out_file<<"\n Here is file ["<<out_data<<"]"
        <<"\n**************\n"
        <<"\n 1) Number of blanks ==>:" <<blank
        <<"\n 2) Number of letters ==>: " <<letter
        <<"\n 3) Number of digits ==>: " <<digit
        <<"\n 4) Number of lines ==>: " <<line;
out_file.close();  // close output file
getch();
}
```

هذا البرنامج سوف يمر بالخطوات التالية عند تنفيذه

1) ظهور الرسالة التالية:-

```
Enter data file name ==>:
```

مطالبة بطباعة اسم الملف الموجود به البيانات المراد معالجتها وهو الملف input.dat والذي سبق تخزين البيانات فيه على هيئة السطور التالية :-

```
This is an example
Using file
Date is 31/12/1999
```

2) يرد الحاسب بالرسالة الموالية:-

```
Enter output data file ==>:
```

طالبا طباعة اسم الملف المطلوب حفظ المطـاليب الجديدة فيه ، حيث تم اختيار الملف تحت اسم output.dat.

3) استعملت جملة while

```
while(in_file.get(ch) , !in_file.eof())
```

وهي تفيد أنه طالما لم يؤشر مؤشر الملف in_file الى النهاية ، نفذ الجمل التالية لجملة while أي قراءة البيانات من الملف input.dat حرفا حرفا عن طريق الدالة get() وبالتالي حساب المطاليب المذكورة في هذا المثال .

4) وحتي تكون الصورة واضحة ، تم كتابة بعض التوضيحات مع النتـائج على الملف الخاص بالمخرجات output.dat .

أخيرا وبعد عملية التنفيذ ، سيكون شكل الملف output.dat عند طباعته عن طريق امر التشغيل (TYPE) من النظام (DOS) مشابها للآتي :-

Here is file [output.dat]

****

1) Number of blanks ==>:6

2) Number of letters ==>:30

3) Number of digits ==>:8

4) Number of lines ==>:3

وهذه الاحصائية مطابقة للبيانات الموجودة في الملف Input.dat

3.12) الملفات الثنائية Binary Files

اضافة الى الملفات النصية توجد الملفات الثنائية حيث يدون فيها قيم الرموز الموجودة في لوحة المفاتيح سواء منها القابلة للطباعة مثل الأرقام والحروف الهجائية أو غير القابلة للطباعة مثل المفاتيح End ، Home ، Del ، Ins وغيرها، في صورتها الأصلية أي استخدام النظام الثنائي (binary system).

1) الكتابة في ملف Writing in a file

مثال 12-6) لتكوين ملف ثنائي تحفظ فيه بيانات تخص رقم الطالب واسمه ودرجته لفصل به عدد من الطلبة، نقدم البرنامج التالي الذي يوضح هذه المهمة.

```
//#include<stdio.h>
#include<conio.h>
#include<iostream.h>
#include<fstream.h>
#define FileName "exam.dat"
```

```
main()
{
	struct
	{
		double num;
		float grade;
		char name[20];
	}exam;
	fstream fpt;
	clrscr();
	fpt.open(FileName,ios::out);
				// read data from screen
	for(int i=1;i<3;i++)
	{
		cout<<"Type id number ==>:";
		cin>>exam.num;
		cout<<"Type garde ==>:";
		cin>>exam.grade;
		cin.ignore(80,'\n');
		cout<<"Type name ==>:";
		cin.getline(exam.name,20);
					// write data in file
		fpt.write((char *)&exam, sizeof(exam));
	}
	fpt.close();
	getch();
}
```

في هذا البرنامج خصص اسم الملف exam.dat للثابت الرمزى FileName عن طريق define ، تلا ذلك الاعلان عن التركيبة exam والتي تضم ثلاثة اعضاء ، العضو الأول num والثاني grade والثالث name ثم فتح الملف exam.dat باستخدام الصيغة (ios::out) للدلالة على الكتابة في هذا الملف ، وباستخدام جملة for تمت قراءة رقم الطالب ودرجته واسمه وكتابة هذه البيانات في الملف exam.dat باستخدام دالة الكتابة write والتابعة لفصيلة الاخراج ostream كالآتي:-

```
fpt.write((char *)& exam, sizeof(exam));
```

حيث الدليل الاول للدالة write من النوع (* char) مع استخدام المؤثر (&) ليشير الـى موقـع التخزيـن ، امـا دليلهـا الثـاني فهـو الدالـة sizeof التـي تـأخذ الشكل:

```
sizeof(exam)
```

فهي قيمة من النوع الصحيح تمثل الحجـم أو الحـيز الـذي تشـغله البيانـات المدخلـة الـى المتغيرات الثلاثـة، المضـاعف num والحقيقـي grade والحرفـي name بالبايت في ذاكرة الحاسب.

عند تنفيذ هذا البرنامج، ستكون عملية ادخال البيانات لعدد اثنين من الطلبة كما يلي:-

```
Type id number ==>:990501
Type garde ==>:83
Type name ==>:Meerath bashir
Type id number ==>:990520
Type garde ==>:79
Type name ==>:Mohamed khairi
```

2) قراءة الملف Reading the File

وحتي يتمكن المبرمج من معرفة صحة بياناته التي تم تخزينها في الملـف الثنائي، ينبغي له كتابـة برنـامج آخـر لقـراءة هـذا الملـف تمهيـدا لطباعتـه أو معالجة محتوياته.

مثال 12-7) البرنـامج التالي مهمته قراءة البيانات التـي تـم حفظهـا فـي الملـف exam.dat وهي الرقم والاسم والدرجة عن طريق البرنامج الذي كتب بالمثـال

(6-12) ومن ثم حساب متوسط درجات الفصل وعرضها مع كل البيانات على الشاشة.

```
#include<conio.h>
#include<iostream.h>
#include<stdlib.h>
#include<fstream.h>
void main()
{
        struct
        {
              double num;
              float grade;
              char name[20];
        }exam;
        fstream fpt;
        clrscr();
        fpt.open("exam.dat ",ios::in);
        if(fpt.fail())
        {
              cerr<<"File could not be opened";
              exit(1);
        }
        float sum=0.0;
                        // print out data
        cout<<endl<<"id number      name    grade"<<endl;
        for(int i=1;i<3;i++)
        {
              fpt.read((char *)&exam,sizeof(exam));
              cout <<exam.num<<"\t\t"<<exam.name<<"\t\t"
                    <<exam.grade<<endl;
              sum+=exam.grade;
        }
        cout<<"The average = "<<sum/2;
        fpt.close();
        getch();
}
```

في هذا البرنامج تمت تهيئة الملف exam.dat للقراءة منه باستخدام الصيغة (ios::in) تلا ذلك التأكد من وجود الملف، وبطبيعة الحال فان هذا الملف موجود مسبقا ، وتمت القراءة من ملف البيانات fpt باستخدام دالة القراءة read التابعة لقناة الادخال istream والتي تشبه دالة write وهي كالآتي:-

```
fpt.read((char *)&exam,sizeof(exam));
```

أخيرا عند تنفيذ هذا البرنامج ، وبدون ادخال أي بيانات ، ستكون النتائج مشابهة للآتي :-

```
id number          name          grade
  990501      Meerath bashir       83
  990520 Mohamed khairi      79
  The average = 81
```

مثال 12-8) البرنامج التالي مهمته ادخال البيانات لمصرف ما، حيث يتم قراءة رقم الحساب acctno وعدد الصكوك checks وقيمة الصكوكamount لعدد M من الزبائن وتخزين هذه البيانات في ملف ثنائي، ثم اظهار البيانات المخزنة على شاشة العرض مع المبلغ الاجمالي لقيم الصكوك.

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
#include<fstream.h>
#define M 3
void main()
{
        struct
        {
            int checks;
            long acctno;
            float amount;
        }person;
```

```
    clrscr();
    fstream bank;
    bank.open("bank.dat", ios::in | ios::out);
    int i=0;
    do
    {
        cout<<"Enter account number ==>:";
        cin>>person.acctno;
        cout<<"Enter number of checks ==>:";
        cin>>person.checks;
        cout<<"Enter amount of check ==>:";
        cin>>person.amount;
                    // write data in the file
        bank.write((char *)&person, sizeof(person));
        i++;
    }while(i<M);
    bank.close();
    float sum=0.0;
    bank.open("bank.dat ",ios::in);
    cout <<" \nAccount number  number of checks "
        <<" amount of checks\n";
    i=0;
    do
    {
        bank.read((char *)&person,sizeof(person));
        cout <<"    "<<person.acctno<<"        "
            <<person.checks<<"     "<<person.amount<<'\n';
        sum+=person.amount;
        i++;
    }while(i<M);
    cout<<"The total amount of checks = "<<sum;
    getch();
    bank.close();
}
```

كالمعتاد، فقد تم اشهار مؤشر الملف bank ومن ثم وقع انشاء وفتح الملف bank. dat كما يلي:-

```
bank. open ("bank. dat", ios:: in | ios:: out);
```

على اساس أنه ملف قابل للكتابة فيه والقراءة منه ويجوز هنا الجمع بين الأمرين أي القراءة والكتابة في نفس الملف، وذلك باستخدام المؤثر المنطقي مع صيغة الاخراج والادخال بالصورة:

ios::in | ios::out

يلي ذلك ادخال رقم الحساب acctno وعدد الصكوك checks وقيمة هذه الصكوك amount للزبون الاول ، وبالتالي كتابة هذه البيانات عن طريق دالة الكتابة write ، حيث استخدم المؤثر (&) قبل التركيبة person ليدل على تخزين رقم الحاسب وعدد وقيمة الصكوك ، ويتكرر هذا حتي نهاية جملة do واذا ما نفذ هذا البرنامج وادخلت البيانات التالية لعدد 3 زبائن مثلا، والتي قد تكون كالآتي للزبون الاول:-

```
Enter account number ==>:1
Enter number of checks ==>:20
Enter amount of check ==>:500
```

يليها البيانات الخاصة ببقية الزبائن كما يلي:-

```
Enter account number ==>:2
Enter number of checks ==>:30
Enter amount of check ==>:600
Enter account number ==>:3
Enter number of checks ==>:40
Enter amount of check ==>:400
```

وحيث ان الملف قد اغلق ، عليه يجب فتحه من جديد بقصد القراءة منه عن طريق الدالة read ومن ثم طباعة الصكوك التي صرفها كل زبون والمبلغ الاجمالي لها على شاشة العرض ، والتي تكون مشابهة للآتي :-

Account number	number of checks	amount of checks
1	20	500
2	30	600
3	40	400

The total amount of checks =1500

3) الاضافة الى الملف Appending to the File

في أحيان كثيرة يتوجب على المبرمج اضافـة بيانـات جديـدة الـى ملفـه أو تصحيح الاخطاء التي قد تقع في بعض بياناتـه أو كلهـا ، ولعمـل هـذا يجـب الوصول الى البيانات المخزنة اولا ومن ثم تغييرها او ابقاؤها كما هي ، ويتم الوصول الى هذا بطريقتين:

الاولى: الوصول التتابعي (sequential access) وفي هـذه الحالـة يتحتم المـرور على بعض السجلات في الملف قبل الوصـول الـى السـجل المطلـوب معالجته ، هذا من ناحية ، ومن ناحيـة اخـرى عنـد اضافـة أي سـجل جديد الى هذا الملف يجب كتابته في آخر الملـف ، والسبب فـي ذلـك هو أن السجلات التي يتكون منها الملف مرتبة ترتيبا تصاعديا حسب التخزين.

الثانية: وتعـرف بـالوصول المباشـر (direct access) أو الوصـول العشـوائي (random access) لسـجل داخـل الملـف، وهـي طريقـة اكـثر ملاءمـة لمعالجة الملفات، وهي تحتاج الى وقت أقل وأيضا يمكن عن طريقهـا اضافة أو الغاء أي سجل في أي موقع من الملف بسرعة كبيرة.

مثال 12-9) وحتي تكون الصورة واضحة عند التعامل مع الملفات الثنائية، نختم هذا الفصل بكتابة برنامج كامل مهمته معالجة البيانات الخاصة بمخزن مبيعات نلخصها في النقاط التالية :-

1) انشاء ملف ليحفظ فيه رقم الصنف واسمه وثمنه لعدد من الاصناف.
2) قراءة هذه البيانات المحفوظة بالملف وطباعتها.
3) التعديل في ثمن الكتاب.
4) امكانية اضافة بيانات جديدة الى الملف.

بما ان هذا المثال يضم مجموعة من المطاليب وكل مطلوب قد يحتاج الي برنامج كامل لوحده، عليه تم تقسيم كل مطلوب خطوة الى عدد من الدوال الفرعية كل دالة تقوم بمهمة معينه.

```
#include <conio.h>
#include <stdlib.h>
#include <iostream.h>
#include <fstream.h>
struct
{
	int ItemNum;
	char name[20];
	float price;
}store={0,"",0.0};

void display();            // دالة عرض قائمة بالاختيارات
void create_file();        // دالة لانشاء الملف
void read_file();          // دالة قراءة البيانات
void update_record();      // دالة التعديل
void insert_record();      // دالة ادخال بيانات جديدة
char filename[20];
char answer;

          //   *******  main function  *******
```

```
main()
{
	int op;
	clrscr();
	do
	{
		display();
		cin>>op;
		switch(op)
		{
			case 1:  create_file(); break;
			case 2:  read_file(); getch(); break;
			case 3:  update_record(); break;
			case 4:  insert_record(); break;
		}
	}while(op>0 &&op<5);
	cout<<"\nGood bye my friend and see you later";
	getch();
	return 0;
}
			//    *******  selection function  *******

void  display()
{
	clrscr();
	cout <<endl
		<<"|------------------- |"<<endl
		<<"| selection list     |"<<endl
		<<"|================== |"<<endl
		<<"| 1) Creat a file    |"<<endl
		<<"| 2) read a file     |"<<endl
		<<"| 3) update a file |"<<endl
		<<"| 4) insert data     |"<<endl
		<<"| 5) exit            |"<<endl
		<<"|------------------- |"<<endl
		<<" enter your selection ==>: ";
}

		//  *******  create a file  *******
void create_file()
{
```

```
        clrscr();
        cout<<"Type your file data [ STORE.DAT] ==>:";
        cin>>filename;
        ofstream filedata("store.dat",ios::out);
        if(filedata.fail())
        {
             cout<<"Error file could not be opened"<<endl;;
             exit(1);
        }
        for(int rec=1;rec<=500; rec ++)
             filedata.write((char *)&store,sizeof(store));
        cout<<"Enter record number (1 to 500) or 0 to stop) ==>:";
        cin>>store.ItemNum;
        while(store.ItemNum>0 && store.ItemNum<=500)
        {
             cout<<"Enter the price ==>:";
             cin>>store.price;
             cout<<"Enter name of item ==>";
             cin>>store.name;
             filedata.seekp((store.ItemNum-1) * sizeof(store));
             filedata.write((char *)&store, sizeof(store));
             cout<<"Enter item number ==>:";
             cin>>store.ItemNum;
        }
        cout<<"\nPress any key to return to the selection list";
}

             //   ******* read data from the file *******
void read_file()
{
        clrscr();
        cout<<"Type your file data [ STORE.DAT] to print out ==>:";
        cin>>filename;
        ifstream InFile(filename,ios::in );
        if(InFile.fail())
        {
             cout<<"Error file ["<<filename<<"] could not be opened";
             exit(1);
        }
        cout<<"   Item number    price    name"<<endl;
        cout<<"*********************************"<<endl;
```

```
        InFile.read((char *)&store,sizeof(store));
        while(!InFile.eof())
        {
            if(store.ItemNum !=0)
            cout <<"\t"<<store.ItemNum<<"\t"
                <<store.price<<"\t"<<store.name<<endl;
            InFile.read((char *)&store,sizeof(store));
        }
        cout<<"\nPress any key to return to the selection list";
}

            //  *******   update record function  *******
void update_record()
{
        clrscr();
        cout<<"Type your file data [ STORE.DAT] to update ==>:";
        cin>>filename;
        fstream updateRec(filename,ios::in | ios::out | ios::nocreate);
        if(updateRec.fail())
        {
            cerr<<"File error"<<endl;
            exit(1);
        }
        do
        {
            int number;
            cout<<"Enter new record number please ==>:";
            cin>>number;
            updateRec.seekg((number-1)* sizeof(store));
            updateRec.read((char *)&store,sizeof(store));
            if(store.ItemNum ==number)
            {
                cout <<endl<<"The old record for item number "
                    <<store.ItemNum<<" as following :"<<endl
                    <<"The item name is "<<store.name<<endl
                    <<"The item price is "<<store.price<<endl<<endl;
                cout<<"Do you like to change the price [Y/N]? :";
                cin>>answer;
                if(answer=='Y' || answer=='y')
                {
                    cout<<"Enter new price ===>:";
```

```
                        cin>>store.price;
                        updateRec.seekp((store.ItemNum-1)*sizeof(store));
                        updateRec.write((char *)&store,sizeof(store));
                    }
                }
                else
                    cerr <<endl<<"Record number "<<number<<" Not found "
                        <<endl;
                    cout <<"Do you like to insert another record [Y/N]?:";
                    cin>>answer;
                }while(answer =='Y' || answer =='y');
                cout<<"\nPress any key to return to the selection list";
}

                    // ******* insert record function *******
void insert_record()
{
        clrscr();
        cout<<"Type your file data [STORE.DAT] to insert new record =>:";
        cin>>filename;
        fstream insertRec(filename,ios::in | ios::out | ios::nocreate);
        if(insertRec.fail())
        {
            cout<<"Error file ["<<filename<<"] could not be opened";
            exit(1);
        }
        ofstream outFile("store.dat",ios::ate);
        if(!outFile)
        {
            cerr<<"File error"<<endl;
            exit(1);
        }
        do
        {
            int number;
            cout<<"Enter new record number please ==>:";
            cin>>number;
            insertRec.seekg((number-1)* sizeof(store));
            insertRec.read((char *)&store,sizeof(store));
            if(store.ItemNum ==0)
            {
```

```
        cout<<"Enter the price ==>:";
        cin>>store.price;
        cout<<"Enter name of item ==>";
        cin>>store.name;
        store.ItemNum=number;
        outFile.seekp((number-1) * sizeof(store));
        outFile.write((char *)&store, sizeof(store));
      }
      else
      cerr <<endl<<"Record number "<<number<<" already exist\n";
      cout<<"Do you like to insert another record [Y/N]?:";
      cin>>answer;
      }while(answer =='Y' || answer =='y');
}
```

للوصول الى البيانات المحفوظة بملف من النوع السابق يستحسن أن تكون هذه البيانات على هيئة تركيبة لتسهيل معالجة اعضاء هذه التركيبة ، وعليه تم استخدام التركيبة store التي تحتوي على ثلاثة اعضاء ItemNum من النوع الصحيح و price من النوع الحقيقي و name من النوع الحرفي مع اعطاء هذه الاعضاء الثلاثة قيما ابتدائية حسب نوعها.

بدُأت الدالة الرئيسية بعرض قائمة الاختيارات باستخدام الدالة display () مع استدعاء مجموعة من الدوال حسب طلب الشخص المنفذ عن طريق جملة switch.

عند تنفيذ البرنامج تمسح شاشة العرض وتظهر قائمة الاختيارات الآتية:

```
|------------------|
| selection list   |
|==================|
| 1) Creat a file  |
| 2) read a file   |
| 3) update a file|
| 4) insert data   |
| 5) exit          |
|------------------|
enter your selection ==>:1
```

يتبعها ادخال الاختيار المناسب وهو الرقم 1 وعليه سوف يتحول التحكم الى الحالة الاولي وهي استدعاء الدالة create_file() بعد كتابة اسم الملف store.dat الذي تحفظ به بيانات المخزن مباشرة بعد الرسالة التالية :-

Type your file data [STORE.DAT] ==>:store.dat

يلي ذلك البحث عن هذا الملف ، وفي حالة عدم وجوده سوف تظهر الرسالة التوضيحية التالية :-

Error file could not be opened

للدلالة على عدم وجوده ، وفي حالة وجود الملف سيتم استخدام جملة for لاعطاء 500 سجل التي يتكون منها الملف قيما فارغة حسب نوعها من خلال دالة الكتابة write بعدها يتم قراءة رقم كل سجل مع ايقاف عملية القراءة بادخال الرقم 0 ، عن طريق لوحة المفاتيح ، وقد تكون الحقول التي يتكون منها السجل الاول كالآتي :-

Enter record number (1 to 500) or 0 to stop) ==>:115
Enter the price ==>:594.25
Enter name of item ==>tables

عندها يحفظ السجل الأول في الملف store.dat عن طريق دالة الكتابة write ويستمر تكرار استقبال باقي السجلات على النحو الموالي

Enter record number (1 to 500) or 0 to stop) ==>:220
Enter the price ==>:367.25
Enter name of item ==>chairs
Enter record number (1 to 500) or 0 to stop) ==>:330
Enter the price ==>:325.75
Enter name of item ==>stepladder
Enter record number (1 to 500) or 0 to stop) ==>:0

بعد ذلك يجرى ايقاف هذا التكرار وانهاء مهمة هذه الدالـة والرجـوع الـى نقطة الاستدعاء بالدالة الرئيسية ، وهي ظهور قائمة الاختيارات السابقة علـى شاشة عرض نظيفة.

واذا رغبنـا فـي عـرض البيانـات التـي تـم ادخالها ، يجـب علينـا ادخـال الاختيار رقم 2، وعليه سوف تستدعي الدالة read_file() لتفي بـالغرض حيث تطـالب هـذه الدالـة باعطـاء اسـم الملـف المطلـوب عـرض محتوياتــه وهــو store.dat وبالتالي قراءة البيانات بالملف عن طريق دالـة القـراءة read طالمـا ان المؤشر لم يصل الى نهاية الملـف ، عمومـا سـوف تكـون شاشـة العـرض كالآتي :-

```
Type your file data [ STORE.DAT] to print out ==>:store.dat
Item number        price        name
*************************************
     115           594.25       tables
     220           367.25       chairs
     330           325.75       setpladder

Press any key to return to the selection list
```

حيث الرسالة الاخيرة " تقول اضغط علـى أي مفتـاح لـلرجوع الـى قائمـة الاختيارات "، وفي حالة الاختيار الثالث وهو التعديـل فـي محتويـات أي مـن السجلات المحفوظة بالملف السابق ، يجرى اسـتدعاء الدالـة update_record() ويكون هناك حوار بين الحاسب والمستخدم ويبدأ المستخدم بادخال اسم الملف الذي يحتوي على البيانات المراد اجراء التعديل عليها

```
Type your file data [ STORE.DAT] to update ==>:store.dat
```

يلي ذلك ادخال رقم السجل 22 مثلا بقصد التعديل فيه بعد ظهور الرسالة التوضيحية التالية :-

Enter new item number please ==>:22

هنا تم استخدام دالة العضو seekg للتأكد من موقع المؤشر عند بداية الملف، وبالتالي تحديد مكان وجود رقم هذا السجل ، ثم استخدام دالة القراءة read والتي مهمتها قراءة الحقول التي يضمها هذا السجل فاذا لم يتم العثور عليه ، تظهر الرسالة

Record number 22 Not found

وتعني ان السجل 22 غير موجود ، اما في حالة اعطاء رقم السجل الموجود اصلا وليكن 220 هنا يتم طباعة حقول هذا السجل وهي السعر والاسم على شاشة العرض كالآتي :-

The old record for item number 220 as following :

The item name is chairs

The item price is 367.25

يلي ذلك اعطاء الحاسب الفرصة للشخص المنفذ بأن يتم الاختيار بين التعديل في حقل السجل وهو السعر من عدمه ، فاذا كان الاختيار بنعم عندها يطبع الحرف Y بعد الرسالة الموالية

Do you like to change the price [Y/N]? :Y

وادخال السعر الجديد وليكن 350 بدلا من السعر القديم وهو367.25 بعد الرسالة:

Enter new price ===>:350

وفي النهاية تظهر رسالة تخطر المنفذ بالتعديل في أي سجل بطباعة الحرف Y أو أيقاف عملية التعديل بطباعة الحرف N مثلا

Do you like to update another record [Y/N]?:N

وبالتالي يجرى الخروج من هذه الدالة والرجوع الى قائمة الاختيارات مرة أخرى.

لاضافة أي سجل جديد الى السجلات المحفوظة بالملف store.dat ، يتم هذا عن طريق الاختيار رقم 4 لكي تستدعي الدالة insert_record() حيث استخدمت دالة العضو seekg لوضع المؤشر في بداية الملف store.dat وبالتالي يجرى البحث عن رقم السجل المراد اضافته باستخدام دالة القراءة read ، وفي حالة العثور عليه يرد الحاسب بوجوب ادخال حقول هذا السجل واستخدام دالة العضو seekp مع دالة الكتابة write لتخزين البيانات في الموقع الصحيح بالملف store.dat ، عموما سوف تكون شاشة العرض عند ادخال السجل الجديد 400 بالشكل المشابه للآتي :-

Type your file data [STORE.DAT] to insert new record ==>:store.dat
Enter new record number please ==>:400
Enter the price ==>:970.75
Enter name of item ==>book

اما في حالة ادخال الرقم الموجود اصلا وليكن 220 سيعطي الحاسب الرسالة :

Record number 220 already exist

وللخروج من هذه الدالة يجب الرد على الرسالة التالية بادخال الحرف N كالآتي :-

Do you like to insert another record [Y/N]?:N

حينها سوف تعود قائمة الاختيارات الى الظهور على شاشة واذا ما ادخل الرقم 2 عندها سوف نشاهد محتويات الملف store.dat موضحا به التعديلات والاضافات الاخيرة ، وهكذا يستمر التحاور بين الشخص المنفذ وجهاز الحاسب بالطريقة السابقة حتي يتم ادخال الاختيار رقم 5 لايقاف تنفيذ هذا البرنامج نهائيا.

4.12) تمارين Exercises

(1) اذكر الفرق بين

(a) Sequential Files (b) Direct Files

(c) Binary Files (d) Text Files

(e) Record (f) Field

(2) المطلوب كتابة برنامج كامل خاص بالادوية فـي صيدليـة مـا بحيث يتم انشاء تركيبة تحت اسم drug تضم الحقول الموضح بتمرين (5) بـالفصل الحادي عشر ، بعد استقبال بيانات هذه الحقول ، المطلوب تخزينهـا فـي ملف تحت اسم pharmacy ، أيضـا انشـاء ملـف آخـر تحـت اسـم الـدواء filedrug يضم اسم الدواء وجهـة الانتـاج والسـعر الكلـي لجميـع الادويـة المنتهية صلاحيتها (حدد تاريخ الانتهاء).

(3) المطلوب كتابة برنامج ينتج عنه الشكل التالي

```
1
123
12345
1234567
12345
123
1
```

مع حفظه في ملف نصي مناسب.

(4) اكتب برنامجا لقراءة وتخزين البيانات التالية في ملـف ثنـائي تحـت اسـم Binary وهي رقم الزبـون والعنـوان وتـاريخ آخـر المشـتريات (الشـهر والسنة) وقيمة الدين ثم اصدار تقرير يحفـظ فـي ملـف آخـر يضـم رقـم

الزبون وعنوانه والدين الذي عليه كذلك باستخدام رقم الزبون، المطلوب التعديل في تاريخ آخر المشتريات وقيمة الدين بالملف الاصلي Binary.

(5) اكتب برنامجا يستقبل عددا من السطور من لوحة المفاتيح، وتخزينها في ملف نصي، ثم كتابة برنامج آخر مهمته قراءة البيانات الموجودة في الملف السابق مع ايجاد وطباعة عدد السطور وعدد الرموز الخاصة وعدد الكلمات المتكون منها كل سطر.:-

(6) اكتب برنامجا لقراءة البيانات المحفوظة في الملفات، الاول به رقم الموظف واسمه ونوع عمله، والثاني به رقم الموظف وراتبه وتاريخ تعيينه.

المطلوب اولا استحداث ملف جديد يضم رقم الموظف ونوع العمل والراتب.

المطلوب ثانيا انشاء ملف آخر يضم قائمة بارقام الموظفين ونوع العمل مع تاريخ التعيين لكل موظف.

(7) المطلوب كتابة برنامج لقراءة البيانات من ملف البيانات الذى اعد في تمرين (2) وينتج الآتي :-

- تقرير كامل بمحتويات الصيدلية من الدواء.
- التعديل في ثمن الدواء عن طريق رقم الدواء .
- تقرير بكل الادوية التي انتهت صلاحيتها للجهة المنتجة ، مع حذف السجل المذكور من الملف الرئيسي.
- اضافة دواء جديد.

(8) اكتب برنامجا يتم فيه ادخال رقم واسم عدد من الطلبة مع المواد التي ينبغي تسجيلها والمبلغ المطلوب من كل طالب مقابل تسجيله، واصدار

تقريرين، الاول يحفظ في ملف يضم الطلبة الذين لم يقوموا بدفع اشتراكاتهم في تسجيل المواد بالجامعة، اما التقرير الثاني والذي يضم ارقام الطلبة والمواد التي تم تسجيلها في ملف آخر.

(9) اكتب برنامجا كاملا ينتج عنه ملف بيانات يخص مصرف الدم ، حيث يتم قراءة البيانات عن طريق دالة فرعية وتخزينها في ملف رئيسي خاص والبيانات هي رقم الشخص واسمه وعنوانه ورقم الهاتف مع العمر وفصيلة الدم، والمطلوب هو:

* طباعة رقم الشخص واسمه ورقم الهاتف للاشخاص الذين لديهم فصيلة الدم A موجبة.
* تخزين الاسم والعنوان والعمر للاشخاص الذين لديهم فصيلة الدم O أو A سالبة ، مع عددهم في ملف آخر.
* انشاء ملف ثالث تخزن فيه كل التعديلات (اضافة والغاء) للبيانات الموجودة في الملف الرئيسي.
* انتاج قائمة بكل الاسماء وارقام الهواتف للاشخاص الذين تكون اعمارهم اقل من 18 سنة ولديهم فصيلة الدم B سالبة أو موجبة.

الفصل الثالث عشر: الفصائل

نظرا للحاجة الماسة التي ظهرت لايجاد ابتكار جديد في برمجة التطبيقات الحديثة ، فقد تم ابتكار اسلوب جديد ليواكب التطور الحاصل في مجال البرمجيات وسرعة صيانتها وتوفير مرونة كبيرة لمبرمجي التطبيقات ومصمميها ، ولقد اعتمد هذا الابتكار بشكل مركز على الهدف (Object) المطلوب الوصول اليه ، لذا فقد سميت اللغة التي انبثقت منه بالبرمجة الموجهة نحو الهدف (Object Oriented Programming) وتختصر بالحروف (OOP) قد تمت اضافة هذا الابتكار الى لغة سي++.

ان اسلوب البرمجة التقليدية يعتمد على البيانات أو الاجراءات (Procedure) التي يكتبها المبرمج بطريقة منفصلة ، وعليه قد تستخدم بيانات خاطئة لهذا الاجراء أو ان بيانات معينة يطبق عليها اجراء مخالف لما هو مطلوب، وقد تم تجاوز هذه المشكلة بالاسلوب الجديد للبرمجة الموجهة نحو الهدف، حيث تعتمد على تمثيل البيانات بطريقة مجردة (Abstract) وتعبئتها مع الدوال الممثلة لسلوك البيانات في كبسولة (Encapsulation) واحدة واضافة خواص اخرى مثل الوراثة (Inheritance) والتي تعني بتوريث صفات وخصائص جديدة علاوة على الصفات الاصلية ، ايضا تعدد الأشكال (Polymorphism) وتعني الشيء الواحد الذي يأخذ اشكالا كثيرة مثل استخدام دالة تحت اسم واحد لتحقيق نتائج متعددة يعتمد على الهدف الذي يستدعي أو ينفذ هذه الدالة.

وفي هذا الفصل نستعرض طريقة أخرى من الطرق التي تتيحها لنا هذه اللغة المهمة وهي طريقة استخدام الفصيلة (Class) ، وسميت بهذا الاسم لأنها

تتشابه مع أية فصيلة أخرى من الفصائل العضوية وذلك من حيث كونها ذات خصائص عامة تشترك فيها كافة الأعضاء المنتمية الى هذه الفصيلـة ، ومـن خلال استعراضنا لهذه الطريقة واستخدامنا لها في التعامل مع البيانات سوف نقدم مفهوما جديدا من مفاهيم البرمجـة يطلـق عليهـا البرمجـة الموجهـة نحـو الهدف، وسوف نعرض مجموعة من الأمثلـة التـي تسـاعد القـارئ علـى شـق طريقه نحو التعرف على المزيد من المزايا التي تشتمل عليها لغة سي++ في هذا المجال والتي تميزها عن بقية لغات الحاسب الآلي الأخرى.

1.13) اعلان الفصيلة Class Declartion

في لغة سي++ يمكن الاعلان عن الفصيلـة التـي هـي عبـارة عـن نوعيـة بيانات المستخدم المعرفة (user defined type) وعن طريقها يتم استحداث عـدد من المتغيرات المختلفة الانواع والتي يطلق عليها اسـم الاهـداف (objects) أو الاعضـاء (members) وعدد من الدوال يطلق عليهـا دوال الاعضـاء (member functions) ، وبالتالي يمكن للمبرمجين المهرة من استحداث وانشاء مجموعـة من الاهداف التي تتناسب مع البيانات وينتج عنها البرنامج المتكامل ، ويمكـن الاعلان عن الفصيلة بالشكل العام التالي

```
class  Class_Name
{
      public :
              Member_Specification_1
              Member_Specification_2
              ...
              Member_Specification_n
      private :
              Member_Specification_n+1
              Member_Specification_n+2
              ...
};
```

حيث:

class كلمة محجوزة بمعني فصيلة.

Class_Name معرف يمثل اسم الفصيلة.

Member_Specification_1 وهو يمثل الجزأ الخاص بالاعضاء الخاصة حيث تستخدم الكلمة المحجوزة (private) قبل هذه الاعضاء والتي لا يمكن استخدامها أو الوصول اليها الا عن طريق دوال الاعضاء أي انها اعضاء محمية من الاستخدام العام .

Member_Specification_n+1 وهو يمثل الجزء الخاص بالاعضاء العامة أي المتاحة للاستخدام العام حيث تستعمل الكلمة المحجوزة (public) قبل هذه الاعضاء.

مثال 1-13) انظر الى الاعلان التالي:-

```
class example
{
        private:   int a,b;
        public :
        void print_out(void)
        {
            . . .
        }
};
```

هنا تم الاعلان عن الفصيلة تحت اسم example التي جاءت بعد كلمة class تلا ذلك قوس بداية الفصيلة example التي تضم مجموعة من المتغيرات b,a ويطلق عليها بيانات الاعضاء (data members) وهي اعضاء خاصة لايمكن الوصول اليها واستخدامها لانها محمية الا عن طريق الدالة العامة public

تحت اسم print_out والتي تسمي دالـة الاعضـاء (members function) أخـيرا يجب انهاء الفصيلة بقوس يتبعه الفاصلة المنقوطة.

2.13) تطبيق الفصيلة Class Implementation

لا يمكن الاستفادة من دوال الاعضاء التي تم الاعلان عنها داخل الفصيلـة الا بتعريفها سواء بعد اسم الفصيلة مباشرة أو بعد الدالة الرئيسية main .

مثال 2-13) لتطبيق الفصيلة أي تعريـف الدالـة print_out التـي لا تسـتقبل أيـة بيانـات ونقـول بانهـا فارغـة void حيـث يتـم تخصيـص القيمتيـن 999,777 للمتغيرين الاعضـاء b,a على التوالي وبالتالي طباعتهما.

```
void print_out(void)
{
        a=777;
        b=999;
        cout<<"A="<<a<<"  B="<<b<<endl;
}
```

3.13) الاعلان عن الاهداف Objects Declaration

يتم الاعلان عن الاهداف باستخدام اسم الفصيلة ثم الهدف ، خذ مثلا الاعلان

```
example point;
```

يعني هنا ان الهدف point اصبح نوعاً من انواع الفصيلة example

4.13) معالجة الاهداف Objects Processing

لكي يتم معالجة الاهداف باستخدام دوال الاعضاء ، يجب استدعاء الدالـة المعنية ويتم ذلك بالصورة التالية:-

```
ObjectName.function()
```

أي اسم الهدف يتبعه مؤثر النقطة (.) يليه اسم دالة العضو.

فمثلا:

```
point.print_out();
```

هنا تم استخدام الهدف تحت اسم point يليه اسم الدالة print_out بدون معاملات وبينهما مؤثر النقطة.

مثال 13-3) البرنامج التالي مهمته توضيح الاعلان عن الفصيلة وكيفية التعامل مع اعضائها وذلك باستدعاء الدالة المكتوبة بالمثال (13-2).

```
#include <iostream.h>
class example
{
	private:   int a,b;
	public :
	void print_out(void)
	{
		a=777;
		b=999;
		cout<<"A="<<a<<"  B="<<b<<endl;
	}
};

main()
{
	example point;
	point.print_out();
	return 0;
}
```

عند تنفيذ هذا البرنامج سينتج عنه السطر الآتي :-

```
A=777   B=999
```

مثال 13-4) في البرنامج التالي يتم قراءة قيمتين من النوع الصحيح ومــن ثـم ارسالهما الى دالـة العضـو وتربيعهمـا والرجـوع بهمـا الـى نقطـة الاسـتدعاء بالدالة الرئيسية.

```
#include <iostream.h>
class point
{
        public :
        int i,j;
        void square(int a,int b)
        {
              i=a*a;
              j=b*b;
        }
};

main()
{
        int num1,num2;
        point two_point;
        cout<<"Enter two integer numbers ==>:";
        cin>>num1>>num2;
        two_point.square(num1,num2);
        cout <<"Square of first number is "<< two_point.i<<endl
              <<"Square of second number is "<< two_point.j;
        return 0;
}
```

تم الاعلان عن الفصيلة تحت اسم point والتي تضم الاعلان عن عضـوين عامين من النوع الصحيح j,i ودالة العضو تحت اسم square التي لها معـاملان من النوع الصحيح b,a والتي مهمتها استقبال قيمتين من الدالة الرئيســية ومـن ثم تربيعهما وتخصيصهما للعضوين j,i والرجوع بهما الــي الدالـة الرئيسية ، اما فيما يخص الدالة الرئيسية فهي تحتوي على الاعلان عن الهدف تحت اسم

two_point وشحن الهدف بقيمة المتغيرين num2,num1 وأخيرا طباعة الناتج باستخدام الامرين

two_point.i

two_point.j

أي اسم الهدف two_point ومؤثر النقطة (.) يليه العضو ، واذا ما نفذ هذا البرنامج وادخلت القيم التالية

Enter two integer numbers ==>: 5 7

سيكون الناتج هو المشابه للآتي:

Square of first number is 25

Square of second number is 49

مثال 5-13) المطلوب كتابة برنامج كامل مهمته الاعلان عن فصيلة تحتوي على دالة عضو واحدة مهمتها استقبال قيمة من النوع الصحيح والرجوع بها الى نقطة الاستدعاء.

```
#include <iostream.h>
class try_this
{
        private :
              int number;
        public:
        int fun(void)
              {
                    cout<< "Type integer value==>:";
                    cin >> number;
                    return number;
              }
}

main()
```

```
{
        try_this  AB;
        int a,b;
        a=AB.fun();
        b=AB.fun();
        cout <<"The first value ==>: "<<a<<endl
             <<"The second value ==>: "<<b<<endl;
}
```

يلاحظ أنه لكي تتم القراءة بدالة العضو ، تم الاعلان عن العضو الخـاص number والدالة العامة fun تحت اسم الفصيلـة try_this ، وبعـد الاعـلان عـن الهدف AB بالدالة الرئيسية تم استدعاء الدالـة fun مقرونـة بـالهدف المطلـوب التعامل معه مع مؤشر النقطة ثم جرت قراءة القيمـة الاولـى بالدالـة والرجـوع بها واسنادها الى المتغير a ثم استدعيت الدالة مرة أخرى ومن ثم اسناد القيمة الثانية للمتغير b وعليه سيكون الناتج بعد ادخال القيمتين كالآتي :-

```
Type integer value==>: 100
Type integer value==>: 500
The first value ==>: 100
The second value ==>: 500
```

5.13) دوال البناء Constructors Functions

تمتاز لغة سي++ بخاصية استخدام دوال بناء الاهداف (Cnstructors) التـي يمكن كتابتها وتعريفها بنفس الطريقة التي تتبع مع دالة العضو حيـث لابـد ان تحمل هذه الدالة نفس اسم الفصيلة وتكون عامة (public) وليس لها نوع محدد أي لا يسبقها كلمة int أو float أو void ، خد مثلا التعريف التالي:

```
example::example(int a,int b)
{
  cout<<"A="<<a<<"  B="<<b<<endl;
}
```

حيث يكتب اسم الدالة وهي example متبوعة باسم الفصيل example وبينهما مؤثر النطاق (::)

مثال 13-6) يمكن التعديل في البرنامج المكتوب بالمثال (13-3) باستخدام دالة البناء.

```
#include <iostream.h>
class example
{
	public :
	example(int a,int b);
};

main()
{
	example example(777,999);
	return 0;
}

example::example(int a,int b)
{
	cout<<"A="<<a<<"  B="<<b<<endl;
}
```

هنا استخدمت دالة البناء example التي تستقبل القيمتين (777,999) وتخصيصهما للمعاملين b,a ومن ثم طباعتهما في هذه الدالة، نلاحظ ايضا انه تم وضع الدالة example بعد الدالة الرئيسية main() حيث استخدم مؤثر النطاق (::) وهي طريقة أخرى لربط هذه الدالة بالفصيلة التابعة لها والتي تأخذ نفس الاسم example وباستخدام مؤثر النطاق واسم الفصيلة فان المترجم يتعرف عليها كدالة بناء ، وعموما فاذا ما نفذ هذا البرنامج فسوف نحصل على النتيجة التالية :-

```
A=777   B=999
```

طريقة أخرى للتعامل مع دوال البناء ، حيث يتم تعريفها بالكامل داخل الفصيلة ، وفي هذه الحالة تستخدم كدالة خطية وبالتالي لا نحتاج إلى استخدام مؤثر النطاق أو اسم الفصيلة. المثال التالي يوضح هذه الكيفية.

مثال 13-7) فيما يلي برنامجا كاملا مهمته استخدام عدد من دوال البناء كل واحدة لها مهمة خاصة.

```
#include <iostream.h>
class  Example
{
        public :
        Example::Example()
        {
                cout<<"This is a constructor function has no parameter"<<endl;
        }
        Example::Example(int i)
        {
               cout <<"This is a constructor function has one "
                      <<"parameter I= "<<i<<endl;
        }
        Example::Example(int i,int j)
        {
               cout <<"This is a constructor function has two "
                      <<"parameters I= "<<i<<" and J= "<<j<<endl;
        }
};

void main()
{
        Example nopar=Example();
        Example onepar=Example(55);
        Example twopar=Example(66,88);
        cin.get();
}
```

تم استخدام عملية الاسناد أو التخصيص (=) لشحن اهداف جديدة بقيم مختلفة ، حيث تم استدعاء دالة البناء الاولي بدون معاملات عن طريق الامر

Example nopar=Example();

حيث نتج عنها ظهور الرسالة التالية:-

This is a constructor function has no parameter

تلا ذلك استدعاء الدالة الثانية التي لها معامل واحد ونتج عنها السطر التالي:-

This is a constructor function has one parameter I= 55

واخيراً جاء دور الدالة الثالثة واعطت الآتي:-

This is a constructor function has two parameters I= 66 and J = 88

مثال 13-8) يقدم البرنامج الموالي فصيلة تحت اسم three والتي عن طريقها يتم استدعاء دالة داخل الفصيلة two أولا ثم استدعاء دالة أخرى داخل one الفصيلة بنفس الاسم one.

```
#include<iostream.h>
class one
{
	public :
		one()
		{
			cout<<"This is a function in class one"<<endl;
		}
};

class two
{
	public :
		two()
```

```
            {
                  cout<<"This is a function in class two"<<endl;
            }
};

class three:two
{
      public :
            one one ;
};

void main()
{
      three ();
}
```

يوجد في الدالة الرئيسية main جملة واحدة وهي استدعاء الدالة تحت اسم three التي عن طريقها استدعيت دالة البناء two الموجودة بالفصيلة two ونتج عنها طباعة السطر

This is a function in class two

تلا ذلك استدعاء الدالة الثانية واسمها one وبالتالي انتجت السطر الموالي

This is a function in class one

مثال 13-9) اكتب برنامجا كاملا مهمته معالجة الاهداف باستخدام دوال الاعضاء حيث يطبع معلومات تخص عدد اثنين من الطلبة تضم رقم الطالب واسمه ودرجته.

```
#include <iostream.h>
#include <string.h>
class student
{
      private:
```

```
        long id;
        char name[30];
        float grade;
    public:
    void information(long pn,char pname[],float test)
    {
        strcpy(name,pname) ;
        id=pn;
        grade=test;
    }
    void print()
    {
        cout <<"Id number ==>:"<<id<<endl
            <<"Name ==>:"<<name<<endl
            <<"Grade ==>:"<<grade<<endl;
    }
};

void main()
{
    student stud1,stud2;
    stud1.information(198005," Nowfel Bashir",86.5);
    stud2.information(198077," Aasum Ahmed",85.5);
    cout <<endl<<"Student one :"<<endl<<"--------------"<<endl;
    stud1.print();
    cout <<endl<<"Student two :"<<endl<<"-------------"<<endl;
    stud2.print();
}
```

في بداية البرنامج اعلن عن الفصيلة student وهي تحتوي على ثلاثة اعضاء خاصة وهي المتغير id ويمثل رقم القيد والاسم name بينما الدرجة يمثلها المتغير grade ، ايضا تحتوي هذه الفصيلة على دالة العضو العامة information التي لها ثلاثة معاملات ومهمتها استقبال البيانات التي تخص الطالب الاول عن طريق هذه المعاملات وبالتالي تخصيصها للمتغيرات الاعضاء الخاصة المختفية ، ونفس الشيء يتم مع البيانات التي تخص الطالب الثاني ويتم هذا الاستدعاء باستخدام اسم الدالة واسم الهدف وبينهما مؤثر

النقطة (.) كما تم شرحه سابقا ، اخيرا جرى طباعة هذه البيانات عن طريق الدالة print() ، وفيما يلي ناتج تنفيذ هذا البرنامج.

```
Student one :
    --------------
Id number ==>:198005
Name ==>: Nowfel Bashir
Grade ==>:86.5
Student two :
    --------------
Id number ==>:198077
Name ==>: Aasum Ahmed
Grade ==>:85.5
```

تسمح لغة سي++ بتقسيم وفصل البرامج الكبيرة الحجم مثل التـي تسـتخدم فيها الفصائل الى قسمين ، الاول يضم الاعلان عن الفصائل ودوال الاعضاء ووضعها في ملف عنوان (header file) بينما يضم القسم الثـاني الاعـلان عن الاهداف ووضعها في ملف المصدر (source file) أي البرنامج الرئيسي ومـن ثم يجرى الربط بينهما عند التنفيذ.

مثال 13-10) يمكن تقسيم البرنامج المكتوب بالمثال السابق الـى جزئين الاول يضم عنوان ملف يخزن في ذاكرة الحاسب تحت اسم student.h ويحتوي الاعـلان عن الفصائل ودوال الاعضاء تحت اسم الفصيلة exam على النحو الموالي:-

```
class exam
{
    private:
        long id;
        char name[30];
        float grade;
```

```
        public:
        void information(long pn,char pname[],float test)
        {
              strcpy(name,pname) ;
              id=pn;
              grade=test;
        }
        void print()
        {
              cout <<"Id number ==>:"<<id<<endl
                    <<"Name ==>:"<<name<<endl
                    <<"Grade ==>:"<<grade<<endl;
        }
};
```

في حين القسم الثاني وهو البرنامج الرئيسي main يمكن حفظه في الذاكرة تحت اسم student.cpp ويضم مكتبة العناوين iostream.h اضافة الى ذلك عنوان ملف student.h السابق ويلي ذلك الاعلان عن الهدفين stud2,stud1 باستخدام اسم الفصيلة المعلن عنها في الملف student.h تحت اسم exam وبالتالي استدعاء الدوال مقرونة بالهدف المناسب والحصول على الناتج.

```
#include <iostream.h>
#include <string.h>
#include "student.h"

void main()
{
      exam stud1,stud2;
      stud1.information(198005,"Nowfel bashir",86.5);
            //member of function
      stud2.information(198077,"Asum Ahmed",85.5);
      cout <<endl<<"Student one :"<<endl<<"--------------"<<endl;
      stud1.print();
      cout <<endl<<"Student two :"<<endl<<"--------------"<<endl;
      stud2.print();
}
```

في حالة تنفيذ هذا البرنامج (القسم الثاني) سوف تظهرعنه نفس النتـائج الموجودة بالمثال السابق.

مثال 11-13) المطلوب من القارىء محاولة فهم وتتبـع البرنـامج الموالـي أولا وتنفيذه على الجهاز ثانيا.

```
#include <iostream.h>
#include <string.h>
class employee
{
        private :
        char name[80];
        double wage;
        public :
        void putname(char *n);
        void getname(char *n);
        void putwage(double w);
        double getwage(double w);
};
        void employee :: putname(char *n)
        {  strcpy(name,n); }
        void employee :: getname(char *n)
        {  strcpy(n,name); }
        void employee :: putwage(double w)
        {  wage=w;  }
        double employee :: getwage(double wage)
        {  return wage;  }
main()
{
        employee emp;
        char name[80]="HAJER AHMED ALI";
        double wage=5500;
        emp.putname(name);
        emp.putwage(wage);
        emp.getname(name);
        cout<<name<<" makes "<<emp.getwage(wage)<<" dinar per year";
        return 0;
}
```

مثال 13-12) بعد حل المسألة بالفصل التاسع مثال (3-9) بالطريقة العادية ، حان الوقت الآن لاستخدام الفصائل ودوال الاعضاء ، لحل نفس المسأله والتي مهمتها كتابة برنامج لقراءة رقم الطالب ودرجته المتحصل عليها في الامتحان وذلك لفصل به 5 طلبة ، مع طباعة رقم الطالب ودرجته مع الفرق بين كل درجة والمتوسط ، حتي يستطيع القارئ المقارنة بين هذه الحلول واختيار الانسب منها.

```
      // this program inputs a sequence of test scores,
      // finds their average and displays the scores together
      // with their defferences from the average.
#include<iostream.h>
const int number_of_student=5;
class point
{
      private : float sum , avg , score[number_of_student];
      long id_no[number_of_student];
      public  : void set_id_gr(long n,float g,int i)
      {
            score[i] = g;
            id_no[i] = n;
      };
      void sum_g(float g){ sum+=g; };
      void avg_g(){ avg=sum/number_of_student; };
      long get_id(int i){ return id_no[i]; };
      float get_g(int i){ return score[i]; };
      float get_df(int i){ return score[i]-avg; };
}

main()
{
      point student;
      float avg,grade;
      long id;
      cout <<"Type id# and score of "<<number_of_student
            <<" students please : "<<endl;
      for(int i=0;i<number_of_student;i++)
```

```
    {
        cout<<"Enter id# and score "<<i+1<<" ==>: ";
        cin>>id>>grade;
        student.set_id_gr(id,grade,i);
        student.sum_g(grade);
    }
    student.avg_g();
    cout<<"\n-----------------------------------------------"<<endl;
        // display id#, scores, and differences from average
    cout.setf(ios::showpoint | ios::fixed);
    cout.precision(1);
    for(i=0;i<number_of_student;i++)
    {
        cout <<endl<<"ID# "<<student.get_id(i)
            <<" score = "<<student.get_g(i)
            <<" different ==>: "<<student.get_df(i);
    }
}
```

بدأ البرنامج بالاعلان عن الفصيلة point وهي تضم عددا من الاعضاء الخاصة ومنها المصفوفة score من النوع الحقيقي لاستعمالها كمخزن لحفظ درجة الطالب ومصفوفة اخرى من النوع الصحيح الطويل تحت اسم id_no لاستعمالها لحفظ ارقام قيد الطلبة كما تضم هذه الفصيلة بعض الدوال العامة منها الدالة set_id_gr التي مهمتها استقبال البيانات التي تخص الطالب عن طريق الجملة

student.set_id_gr(id,grade,i);

تم تخصيص هذه البيانات وهي الرقم والدرجة للمتغيرات الخاصة في الأماكن التي يشير اليها الدليل i في المصفوفة المناسبة ، بعد حساب المتوسط عن طريق دالة العضو avg_g() ، جرى استعمال جملة for لاخراج البيانات التي تم ادخالها سالفا مصحوبة بالفارق بين درجة كل طالب ومتوسط

الفصل. وعموما اذا ما نفذ هذا البرنامج فسيعطي النتائج المشابهة بنفس المثال المشار اليه.

مثال 13-13) هذا البرنامج يعالج بيانات خاصة ببضاعة محفوظة في مخزن معين لا يزيد عددها على 100 صنف ، حيث تتم قراءة رقم وسعر هذه البضاعة وتخزينها في مصفوفة مناسبة ، ومن بعد يجرى ادخال أي رقم والبحث عنه داخل المصفوفة حيث يطبع الرقم وسعر البضاعة في حالة وجوده، أو تطبع الرسالة المناسبة في حالة عدم وجوده.

```
#include<process.h>
#include<iostream.h>
#include<conio.h>
const LIMIT=100;

                    // Linear search of an array
class point
{
        private : int item[LIMIT] ;
                float  price[LIMIT];
        public  : void set_no(int i, int a)
        { item[i] = a; };
        int get_no(int j)
        {return item[j]; };

        void set_pr(int k, float b)
        { price[k] = b; };

        float get_pr(int m)
        {return price[m]; };
};
main()
{
        point store; int item_no; float item_price; int size;
        clrscr();
        cout<<"Enter size of list price ==>: ";
        cin>>size;    // get size of items
```

```
    if (size>100) exit(1);
    for(int i=0;i<size;i++)
    {
        cout<<endl<<"Enter item number   ["<<i+1<<"] ==>: ";
        cin>>item_no;
        store.set_no(i,item_no);
        cout<<"Enter price of item ["<<i+1<<"] ==>: ";
        cin>>item_price;
        store.set_pr(i,item_price);
    }
    cout <<endl<<"Enter item number to be "
        <<"searched ( OR 0 to QUIT ) ==>: ";
    cin>>item_no;
    int j,found;
    while(item_no != 0)
    {
        j = 0;
        found = 0;
        do
        {
            if(item_no == store.get_no(j))
                found = 1;
            else
                j+=1;
                }while((j < size) && (found != 1));
            if(found == 1)
            {
                cout <<endl<<"The item number "<<item_no
                    <<" has the price ==>: "<< store.get_pr(j);
            }
            else
            {
            cout <<endl<<"Sorry item number "<<item_no
                <<" not found ";
        }
        cout <<endl<<"Enter item number to be "
            <<"searched ( OR 0 to QUIT) ==>: ";
        cin>>item_no;
    }
}
```

بما ان هـذا البرنـامج قد يتطلـب ادخـال ومعالجـة بيانـات كثيرة عليـه تـم الاعلان عن الفصيلة تحت اسم point وتضم العضوين من المصفوفة حيث:

1) الاولي تحت اسم item من النوع الصحيح لتخزين رقم البضاعة.
2) الثانية تحت اسم price من النوع الحقيقي لتخزين سعر البضاعة.

وتضم ايضا عددا من دوال الاعضاء.

عند تنفيذ البرنامج تظهر على شاشة العرض الرسالة التالية طالبـة ادخـال عدد الاصناف بالمخزن وليكن 3 يليه ادخال رقم الصنف وسعره كالآتي:-

```
Enter size of list price ==>: 3

Enter item number [1] ==>: 111
Enter price of item [1] ==>: 55.25

Enter item number [2] ==>: 835
Enter price of item  [2] ==>: 10.99

Enter item number [3] ==>: 321
Enter price of item [3] ==>: 759.75
```

يأتي السؤال عن ادخال الرقم المراد البحث عنه أو ادخال الرقم 0 لانهـاء تنفيذ البرنامج ، ولاختبار هذا البرنامج تم ادخال رقـم الصنـف 111 المحفـوظ سابقاً للبحث عن ثمنه كالآتي:-

```
Enter item number to be searched(0 to QUIT) ==>: 111
```

هنا يقوم البرنامج بالرد على السؤال السابق حيث يطبع رقـم الصنـف 111 والسعر 55.25 التابع له على النحو التالي :-

```
THE ITEM NUMBER 111 HAS THE PRICE 55.25
```

وفي حالة اعطاء الحاسب الرقم 312 بدلا من الرقم الصحيح 321 كالآتي:-

Enter item number to be searched(0 to QUIT) ==>: 312

يأتي الرد مباشرة بأن يطبع البرنامج الرقم والرسالة الدالة على عدم وجوده ضمن الاصناف كالآتي:

SORRY ITEM NUMBER 312 NOT FOUND

يلي ذلك المطالبة بادخال الرقم المراد البحث عنه مرة ثانية، وليكن 321:

Enter item nymber to be searched(0 to QUIT) ==>: 321

عندها يكون الرد هو السطر التالي:

THE ITEM NUMBER 321 HAS THE PRICE 759.75

وهكذا يستمر الحوار حتي يتم ادخال الرقم 0 لانهاء تنفيذ البرنامج، كلآتي

Enter item number to be searched(0 to QUIT) ==>: 0

مثال 13-14) الهدف من البرنامج التالي هو قراءة عدد من القيم الصحيحة، بحيث لاتزيد على 50 قيمة، مع تخزين هذه القيم في مصفوفة ثم ايجاد القيم الموجبة وحفظها في مصفوفة أخرى واخيرا طباعة هذه المصفوفة.

```
#include<iostream.h>
#include<process.h>
const int array_size=50;
class array
{
        private :  int  matrix[array_size] ;
                   float  pos_array[array_size];
        public  :  void set_mat(int i, int d)
                   { matrix[i] = d; };

        int get_mat(int i)
        {return matrix[i]; };
```

```
        void set_positve(int i, int d)
        {pos_array [i] = d; };

        int get_ positve(int i)
        {return pos_array[i]; };
};

main()
{
        array point;
        int n,mat,counter=0;
        cout <<"Enter size of the array "
             <<"less than " <<array_size<<" ==>: ";
        cin>> n;
        cout <<endl<<"Please type "<<n<<" elements now : "<<endl;
        for(int i=0;i<n;i++)
        {
             cout<<"MATRIX ["<<i<<"] ==>: ";
             cin>> mat;
             point.set_mat(i, mat);
        }
        for(i=0;i<n;i++)
        if(point.get_mat(i) > 0)
        {
             point.set_ positve(counter, point.get_mat(i));
             counter+=1;
        }
        if(counter==0)
        {
             cout<<endl<<"Sory no positive value ";
             exit(0);
        }
        cout<<endl<<"The array looks like : ";
        for(i=0;i<n;i++)
        cout<<endl<<"MATRIX ["<<i<<"] = "<< point.get_mat(i);
        cout<<endl<<endl<<"While the positive elements as following :";
        for(i=0;i<counter;i++)
        cout<<endl<<"MATRIX ["<<i<<"] = "<< point.get_ positve(i);
}
```

تنفيذ البرنامج وبعد ادخال حجم المصفوفة وليكن 6 كالاتي:

Enter size of the array less than 50 ==>:6

وعدد عناصرها

Please type 6 elements now :
MATRIX [0] ==>: 22
MATRIX [1] ==>:-6
MATRIX [2] ==>: -8
MATRIX [3] ==>: 44
MATRIX [4] ==>:77
MATRIX [5] ==>: 11

سوف ينتج عنه الناتج التالي:-

The array looks like :
MATRIX [0] ==>: 22
MATRIX [1] ==>:-6
MATRIX [2] ==>: -8
MATRIX [3] ==>: 44
MATRIX [4] ==>:77
MATRIX [5] ==>: 11

While the positive elements as following :
MATRIX [0] ==>: 22
MATRIX [1] ==>:44
MATRIX [2] ==>: 77
MATRIX [3] ==>: 11

بعد الاعلان عن الفصيلة واعضائها الخاصة وهـي مصفوفتـان مـن النـوع الصحيح ودوال الاعضاء العامـة ومنهـا التـي مهمتهـا اسـتقبال القيـم المدخلـة

وتخزينها في المصفوفة matrix ومنها التي تستقبل القيم الموجبة فقط وتخزينها في المصفوفة pos_array بعد ذلك جاء استخدام جملة for الأولي لادخال القيم وتخزينها في المصفوفة matrix، وجملة for الثانية التي خصصت لايجاد القيم الموجبة في المصفوفة السابقة عن طريق الجملة الشرطية if وهو point.get_mat(i) > 0 فإذا تحقق هذا الشرط، عندها نقول ان قيمة العنصر موجبة ويجرى تخزينها في مصفوفة جديدة تحت اسم pos_array عن طريق الدليل counter او ما يسمي بالعداد بداية من العنصر الاول وهو العدد 0 مع اضافة الرقم 1 الى هذا العداد، وذلك لتحديد عدد عناصر المصفوفة pos_array بعدها يتم استخدام هذا العداد في عملية الطباعة، اما اذا كان الشرط خاطئا، فسوف لن يحدث أي تغير في قيمة العداد counter وتبقي قيمته صفرا عندها نقول لا توجد اية اعداد موجبة في المصفوفة matrix وعليه وجب اصدار الرسالة المناسبة والخروج من البرنامج نهائيا.

(6.13) المؤشرات والفصائل Classes and Pointers

سبق أن أوضحنا عند شرحنا لموضوع التراكيب امكانية استخدام المؤشرات (Pointers) للاشارة الى تلك التراكيب للوصول الى الدوال المختلفة، نفس الأمر ممكن بالنسبة للفصائل حيث تسمح لنا لغة سي++ بالتوصل الى الفصائل بطريقة مشابهة على النحو التالي :-

```
PointerName ->member
```

حيث (->) مؤثر التوصل بينما member العضو قد يكون دالة العضو أو متغير العضو.

مثال 13-15) البرنامج الموالي هو اعادة للبرنامج المكتوب بالفصل الحادي عشر مثال (11-15) والذي مهمته حساب المجموع الكلي للاصناف المخزنة في عدد من المخازن وذلك باستخدام المؤثر (<-) مع الفصيلة عوضا عن استخدامه مع التركيبة ولمعرفة الفرق بينهما في الاستخدام.

```
#include<string.h>
#include<iostream.h>

class class_type
{
	private:
		int ItemNum;
		char ItemName[20];
		int store[8];
	public:
		void set_int(int d){ItemNum=d;};
		void set_name(char *ch) {strcpy(ItemName,ch);};
		void set_arr(int element,int pos){store[pos]=element;};
		int get_int(void){return ItemNum;};
		char *get_name(void){return ItemName;};
		int get_arr(int pos){return store[pos];};
		int sum_of_n_numbers(class_type *);
};

int class_type::sum_of_n_item(class_type *q)
{
	char *c;
	int num,sum=0;
	cout << "\nType item name ==>:";
	cin.getline(c,80);
	q->set_name(c);
	cout << "\nType number of store you have ==>:";
	cin>>num;
	cout<<"\n--------------------------------\n";
	q->set_int(num);
	for(int i=0;i<q->get_int();i++)
	{
		cout<<"Enter number of items in store " << i+1<<" ==>:" ;
```

```
            cin>>num;
            q->set_arr(num,i);
            sum+=q->get_arr(i);
        }
        return (sum);
}
void main(void)
{
        class_type store;
        char c;
        int sum=0,num;
        sum=store. sum_of_n_item(&store);
        cout << "\nYou have item name (( " <<store.get_name()<<" ))"
             << "\nin each of the following " <<store.get_int()<<" stores";
        for(int i=0;i<store.get_int();i++)
        cout << "\n store [" << i+1 << "] ==>" <<store.get_arr(i);
        cout <<"\n\nThe total items you have ==>: "<< sum;
        getch();
}
```

في بداية هذا البرنامج تم الاعلان عن الفصيلة تحت اسم class_type والتي تضم ثلاث اعضاء وهي المتغيرات الأعضاء التالية:-

المتغير ItemNum من النوع الصحيح.

المتغير ItemName من النوع الحرفي.

المتغير store نوع المصفوفة الاحادية وحجمها 8 عناصر من النوع الصحيح.

أيضا تعريف الدالة الفرعية sum_of_n_item وتمرير المؤشر الى الفصيلة كمعامل للدالة وهو المتغير q من النوع المؤشر وذلك بأن سبقه المؤشر(*) وبالتالي فهو يعتبر من نوع الفصيلة calss_type وعليه يتم التعامل مع هذا المؤشر باستخدام مؤثر التوصيل (<-) في هذه الحالة ، ومهمة هذه الدالة استقبال البيانات وطباعتها مع حساب مجموع الاصناف والرجوع بالنتيجة وطباعتها في الدالة الرئيسية، وقت تنفيذ البرنامج سيعطي النتائج المشابهة بنفس المثال المشار إليه.

7.13) تمارين Exercises

(1) اشرح ما يقصد بالفصيلة ، مع اعطاء بعض الامثلة على ذلك.

(2) عرف فصيلة تضم الاعضـاء الآتيـة: اسـم الكتـاب، اسـم المؤلـف، ثمـن الكتاب وتاريخ النشر.

(3) المطلوب تصميم فصيلة company تخص مؤسسة الكهربـاء تحتـوي اسـم المستهلك ورقم العداد والقراءة السابقة والحالية مع الديون لعدد num مـن المستهلكين مع ايجاد قيمة الفاتورة عن طريق دالة مناسبة حيث:

قيمة الفاتورة = الاستهلاك × سعر الكيلوات + الديون.

الاستهلاك = القراءة الحالية - القراءة السابقة.

سعر الكيلو = { 20 درهما اذا كان الاستهلاك اقل من 500 كيلوات
{ 25 درهما اذا كان الاستهلاك 500 كيلوات أو اكثر

وطباعة رقم العداد والاسم لاعلى واقل فاتورة باستخدام دالة أخرى.

(4) اعد كتابة البرامج التالية باستخدام الفصائل.

(a)

```
#include<iostream.h>
main()
{
        union example
        {
              char *x;
              char *y;
              char *z;
        }sample;
        sample.x="GREEN";
```

```
        cout <<"X ==>"<<sample.x<<"  Y ==>" <<sample.y
             <<"  Z ==>"<<sample.z<<endl;
        sample.y="BLUE";
        cout <<"X ==>"<<sample.x<<"  Y ==>" <<sample.y
             <<"  Z ==>"<<sample.z<<endl;
}
```

(b)

```
#include<iostream.h>
struct numbers
{
        int i;
        float f;
        double d;
};

print_data(numbers sample)
{
        cout <<"A ==>:"<<sample.i<<"  B ==>:"<<sample.f
             <<"  C ==>:"<<sample.d;
}

void main()
{
        struct numbers ss={10,5.6,0.0012};
        print_data(ss);
}
```

(5) اكتب برنامجاً يستقبل عدداً من السطور من لوحة المفاتيح باستخدام الفصيلة ثم اوجد عدد الرموز الخاصة مع عدد الكلمات التي تبدأ بالحرف B وطباعة محتويات كل سطر بالعكس.

(6) اكتب برنامجا لاشهار فصيلة تضم الاعضاء رقم الموظف واسمه وعدد ايام اجازته مع الراتب لعدد محدد من الموظفين ، والمطلوب الحصول

على تقرير يضم كل البيانات المدخلة مع طباعة البيانات التي تخص الموظف الذي له أعلى راتب.

(7) اعد كتابة البرنامج بتمرين (8) بالفصل الحادي عشر باستخدام الفصائل.

(8) تتبع البرامج التالية ودون ما يطبع باستخدام الورقة والقلم ، ثم قم بتنفيذها على جهاز الحاسب وقارن النتائج في كلا الحالتين.

(a)

```
#include<iostream.h>
class TWO
{ public:
        int i;
        char a;
        void print_i()
        {  cout<<"Value of I==>"<<i<<endl; }
        void print_a()
        {  cout<<"Value of A==>"<<a<<endl; }

};
void main()
{
            TWO *abc;
            abc->i=100;
            abc->a='?';
            abc->print_i();
            abc->print_a();
}
```

(b)

```
#include<iostream.h>
#include <math.h>
class point
{
        public :
        int i,j;
```

```
        void print(int a,int b)
        {
              i=sqrt(a);
              j=pow(i,b);
        }
};

main()
{
        point out;
        out.print(25,3);
        cout<<"First value ==>"<<out.i<<endl;
        cout<<"Second value ==>"<<out.j;
        return 0;
}
```

(c)

```
#include<iostream.h>
const int array_size=10;
class array
{
        private : int  matrix[array_size] ;
        float  pos_array[array_size];

        public  : void set_m(int i, int d)
              { matrix[i] = d; };

              int get_m(int i)
              { return matrix[i]; };
};

main()
{
        array MAT;
        int dig[array_size]={8,0,1,-4,5,2,0,-6,9},pos,neg,zero;
        pos=neg=zero=0;
        for(int i=0;i<array_size;i++)
        MAT.set_m(i,dig[i]);
        for(i=0;i<array_size;i++)
```

```
    if(MAT.get_m(i) > 0)
        pos++;
    else
    if(MAT.get_m(i) < 0)
        neg++;
    else
        zero++;
    cout<<endl<<"Your array looks like : ";
    for(i=0;i<array_size;i++)
    cout<<endl<<"MATRIX ["<<i<<"] = "<<MAT.get_m(i);
    cout<<endl<<"Number of positive value is "<<pos;
    cout<<endl<<"Number of negative value is "<<neg;
    cout<<endl<<"Number of zero value is "<<zero;
}
```

الفصل الرابع عشر الرسومـــات

الحاسب الآلي كما ذكرنا سابقا يتعامل مع مجموعة من البيانـات ثم يقوم بمعالجتها وبالتالي اظهار البيانات والمعلومات التي قد تكون على هيئة ارقـام وحروف ورموز أو قد تكون على هيئة اشكال ورسومات هندسية الي الوسط الخارجي بوسائل مختلفة ، منها شاشة العرض والطابعة والراسمه وغيرها.

لهذا سوف نتنـاول بعـون اللـه موضـوع تعـامل الحاسـب مـع البيانـات واظهارها على شكل صور أو رسومات بيانية تساعد القـارئ علـى الاسـتفادة من الحاسب في هذا المجال.

1.14) اساسيات الرسم Fundamentals of Graphics

حتي يمكننا كتابة برنـامج كـامل يقـوم بمهمـة الرسـم (Graphics) باشـكال معينه، يجب استعمال الشاشات الملونـة (Color Monitors) التـي تسـاعد علـى اظهار الرسومات بمختلف الوانهـا ، وحتـي يمكـن عـرض الرسـومات علـى الشاشة يجب تدعيمها بدائرة الكترونية خاصة تسـمي سـواقة الرسـم (Graphic driver) وتختلف دقة الرسم من شاشة الى أخرى حيث تقاس هذه الدقـة بعـدد النقاط الضوئية التي تسمي بكسل (Pixel) وهي اصغـر نقطـة يمكـن اضاءتهـا على الشاشة حيث يتم تغطية الشاشة بعدد كبير من النقاط افقيا ورأسيا ، ومـن انواع سـواقة الرسـوم المعروفـة CGA التـي تعتـبر مـن اقـدم سـواقات الرسـم الملون وصممت لاظهار عدد من الالوان في حدود 16 لونا وتصل دقتها الـى

(640×200) نقطة أي 200 بكسل بطول الشاشة و640 بكسل بعـرض الشاشـة ، توجد انواع أخرى من سواقة الرسم ومنهـا EGA التـي تعطـي فـي حـدود 64 لونا وبدقة تصـل (640×350) نقطـة للشاشـة الواحـدة أيضـا VGA التـي تصـل دقتها الى (640×480) نقطة على الشاشة.

2.14) انماط الرسم Graphic Modes

فيما يلـي جـدول يبيـن انمـاط الرسـم (Graphic Modes) مـع سـواقة الرسـم (Graphics driver) والمستخدمة في مجال الرسومات :-

Graphics driver	Graphics mode(s)	Key value	Column × Row	Pallet or codes
CGA	CGAC0	0	320 × 200	C0
	CGAC1	1	320 × 200	C1
	CGAC2	2	320 × 200	C2
	CGAC3	3	320 × 200	C3
	CGAHI	4	640 × 200	2 color
MCGA	MCGAC0	0	320 × 200	C0
	MCGAC1	1	320 × 200	C1
	MCGAC2	2	320 × 200	C2
	MCGAC3	3	320 × 200	C3
	MCGAMED	4	640 × 200	2 color
	MCGAHI	5	640 × 480	2 color
EGA	EGA64LO	0	640 × 200	16 color
	EGA64HI	1	640 × 350	16 color
VGA	VGALO	0	640 × 200	16 color
	VGAMED	1	640 × 350	16 color
	VGAHI	2	640 × 480	16 color

ملاحظة: سـواقة وانمـاط الرسـم معتمـدة علـى القيـم العدديـة وقيـم الانمـاط (modes) وعدد الاعمدة الصفوف واللون المشار اليه C0,...,C3 الذي يشير الى اشكال ذات اربعة الوان.

3.14) انماط دوال النص Text Mode Functions

توجد بعض دوال النص التي تستخدم مونوكروم (Monochrome) مثل (mode 7) لعرض نمط النص و (mode 3) لعرض الرسومات ، ولكي تستخدم هذه الدوال نحتاج الى الملف <conio.h> ، والغاية الرئيسية لدوال نمط النص هو امكانية استخدام النوافذ (windows) لتسهيل التعامل مع النص في النافذة.

مثال 14-1) البرنامج الموالي يبين استخدام النافذة ومجموعة من دوال نمط النص والتي تهيء لنا النافذة وتكرر كتابة كلمة في النافذة حتى تمتليء النافذة والكلمات الاضافية تتسبب في تحريك محتويات النافذة الى الاعلى.

```
#include<conio.h>
#include<dos.h>  // for delay function
#define LEFT  10
#define TOP  8
#define RIGHT  52
#define BOT  21
main()
{
      window(LEFT,TOP,RIGHT,BOT);
      textcolor(RED);
      textbackground(GREEN);
      for(int i=0;i<56;i++)
      {
           cputs("Turbo C++ ");
           delay(200);
      }
      getche();
      return 0;
}
```

الشرح:

في هذا البرنامج استخدمت بعض دوال نمط النص وهي :-

1) دالة النافذة ()window تعرف منطقة الشاشة على انها نافذة نص text window حيث يتم تخصيص مساحة من الشاشة كنافذة عن طريق الدالة التي تأخذ الشكل التالي :-

```
window(x1,y1,x2,y2);
```

حيث y1,y1 احداثيات الجزء العلوي ناحية اليسار من النافذة ، في حين ان x2,y2 احداثيات الجزء السفلي ناحية اليمين من النافذة .

فمثلا الدالة (window(1,1,80,25 تعني تكوين نافذة مستطيلة ابعادها تبدأ من العمود 1 والسطر 1 بطول 80 عمودا من الناحية اليمني وبعرض 25 سطرا الى اسفل الشاشة ، وبالطبع يجب الأخذ في الاعتبار ان مدى الشاشة هو 80 عمودا و 25 سطرا .

2) الدالة ()textcolor تختص بلون الحروف المراد كتابتها على النافذة وهو اللون الاحمر وقد تأخذ رقما صحيحا مداه من 0 الى 15 بين القوسين ليمثل نوع اللون الموجود بلغة سي++ حيث يوجد عدد من الالوان التي يمكن إستغلالها في كثير من البرامج ، والتي يوضحها الجدول بالملحق (3) .

3) الدالة (textbackground(GREEN مهمتها اعطاء اللون الاخضر لارضية النافذة الرئيسية التي تم تكوينها بالامر ;(window(LEFT,TOP,RIGHT,BOT انظر إلى جدول الالوان ملحق (3)، حيث تمت كتابة الكلمة Turbo C++ عدد 56 مرة داخل هذه النافذة عن طريق دالة ()cputs التي مهمتها اظهار أي سلسلة حرفية على شاشة العرض في فترات متقطعة باستخدام الدالة ()delay ، وعلى القارئ تنفيذ هذا البرنامج وملاحظة ما ينتج عنه.

4.14) امثلة عامة Genaral Examples

وحتي يمكن كتابة برنامج وبالتالي السماح باستعمال سواقة وانمـاط الرسم وعدد من الدوال ذات الصلة بالرسومات ، يجب استخدام ملف الرسومات <graphics.h> وفيما يلي عدد من الامثلة التي تبين طريقة استخدام بعض الدوال الخاصة بالرسومات.

مثال 14-2) في البرنامج الموالي نقدم بعض الاساسيات التي تساعد المبرمج على الاستفادة من الدوال الجاهزة التي تشتهر بها لغة سي++ والذي مهمته رسم مجموعة من المستطيلات (Rectangles) المختلفة الاشكال.

```
#include <graphics.h>
#include <stdlib.h>
#include <conio.h>
main()
{
        int gdriver = DETECT, gmode, errorcode;
        int left, top, right, bottom;
                // initialize graphics
        initgraph(&gdriver, &gmode, "c:\\tc\\bgi");
                // read result of initialization
        errorcode = graphresult();
        if (errorcode != grOk)  // an error occurred
        {
            cout <<"Graphics error: "<<grapherrormsg(errorcode)
                <<"Press any key to halt:";
            cin.get();
            exit(1); // terminate with an error code
        }

        left = getmaxx() / 2 - 200;
        top = getmaxy() / 2 - 200;
        right = getmaxx() / 2 + 200;
        bottom = getmaxy() / 2 + 200;
                    // draw a rectangle
        setbkcolor(11);
        for(int i=1;i<5;i++)
```

```
	{
		setcolor(i);
		rectangle(left,top,right,bottom);
		outtextxy(250,250,"TEST GRAPHICS");
		outtextxy(250,450," This is a rectangle");
		cin.get();
		cleardevice();
	}
	closegraph();
	return 0;
}
```

يبدأ هذا البرنامج بالاعلان عن gdriver سواقة الرسم graphics-drivers التي تحدد نوع الرسم وقد خصصت لها DETECT أو القيمة 0 وهي تشتغل اتوماتيكيا وقد يخصص لها أي نوع من سواقة الرسم مثل CGA أو EGA أو VGA وغيرها ، اما فيما يخص gmode فهي تحدد نوع الالوان ، يلي ذلك تهيئة وتحميل سواقة الرسم من القرص بحيث يكون النظام system جاهز للرسم عن طريق الامر

```
initgraph(&gdriver, &gmode, "c:\\tc\\bgi");
```

فاذا كان التحميل غير صحيح يجرى اسناد القيمة 0 الى المتغير الصحيح errorcode وبالتالي طباعة نوع الخطأ مع الرسالة المناسبة والخروج نهائيا من البرنامج ، أما اذا كان غير ذلك فيتم ايجاد أو تحديد اكبر قيمة لاحداثيات النقطة x عن طريق الدالة getmaxx() التي ترجع بالقيمة 319 بكسل واكبر قيمة لاحداثيات النقطة y عن طريق الدالة getmaxy() التي ترجع بالقيمة 199 بكسل عند استخدام CGA والقسمة على 2 لتحديد منتصف الشاشة مع طرح القيمة 200 واسناد الناتج للمتغيرين الصحيحين left و top عن طريق الجملتين:

```
left = getmaxx() / 2 - 200;
top = getmaxy() / 2 - 200;
```

وبنفس الطريقة يتم الحصول على قيمة المتغيرين right و bottom كالآتي:

```
right = getmaxx() / 2 + 200;
bottom = getmaxy() / 2 + 200;
```

وذلك لاستعمال هذه المتغيرات في اظهار ورسم مستطيل عن طريق الدالة rectangle التي لها الشكل التالي :-

```
void rectangle(int left,int top,int right,int bottom);
```

حيث المعاملان الأولان left و top يمثلان الزاوية العليا من الناحية اليسرى للشاشة ، بينما المعاملان الاخيران right و bottom يمثلان الزاوية السفلى من الناحية اليمني للشاشة .

قبل البدأ في عملية الرسم تمت تهيئة وتنظيف شاشة العرض واعطاؤها اللون الخلفي المناسب الذي يمثله الرقم 11 عن طريق الدالة التالية:

```
setbkcolor(11);
```

استخدمت جملة for التي تضم الجملة المركبة التي تحتوي على:

1) دالة تلوين خطوط المستطيل وهي setcolor(i) التي تأخذ عدد اربعة الوان حسب قيمة المتغير i.

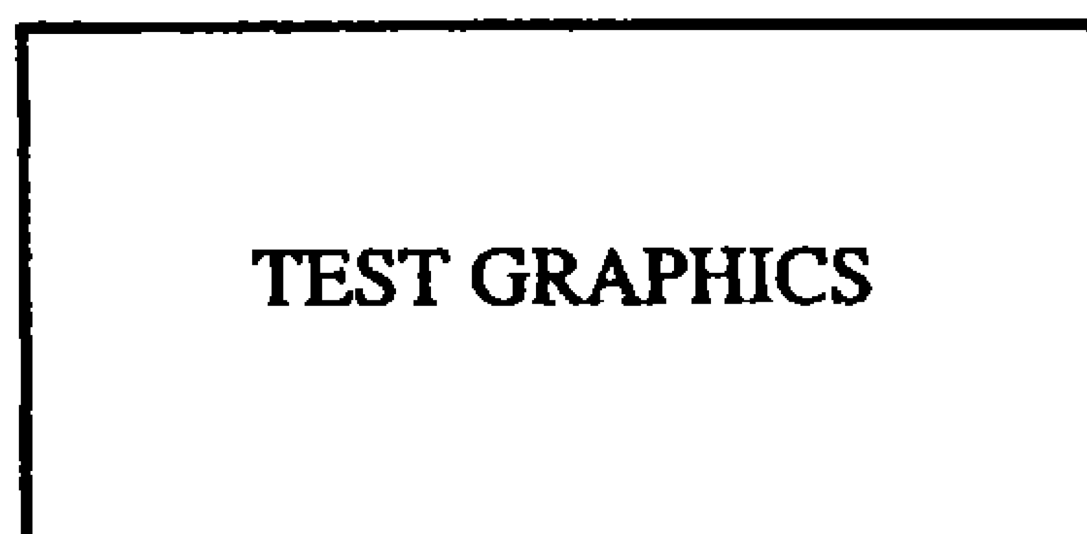
2) دالة رسم اضلاع المستطيل الذي تحدده الزاوية العليا من الناحية اليسري والزاوية السفلي من الناحية اليمني لشاشة العرض .

3) الدالة outtextxy التي تأخذ الشكل التالي:-

```
void outtextxy(int x,int y,char *textstring);
```

ومهمتها اخراج الجملة أو العبارة textstring في المكان المحدد بقيمتي المتغيرين y,x على الشاشة ، حيث x تمثل النقطة على العمود و y النقطة

على السطر أي اخراج الجملة TEST GRAPHICS في هذا البرنامج بمنتصف المستطيل في النقطة (250,250) والجملة This is a rectangle في النقطة (250,450) ومكانها تحت شكل المستطيل ، وعند ظهور شكل المستطيل الاول وبالضغط على مفتاح الادخال (Enter) يتم تنظيف الشاشة مرة أخرى عن طريق الدالة cleardevice() ويظهر المستطيل الثاني وهكذا يتكرر تنفيذ هذه الجمل حتي نهاية حلقة التكرار عندها يقفل النظام الخاص بالرسم عن طريق الدالة closegraph() ، واذا ما نفذ هذا البرنامج فسينتج عنه رسم المستطيل المشابه للآتي :-



This is a rectangle

مثال 3-14) البرنامج الموالي مهمته القيام برسم دائرة (Circle) ذات اقطار (Radius) مختلفة.

```
#include <graphics.h>
#include <stdlib.h>
#include <conio.h>
main()
{
        int gdriver = DETECT, gmode, errorcode;
        int xradius = 100;
        int x = getmaxx() / 2 + 280;
        int y = getmaxy() / 2 + 280;
                // initialize graphics
        initgraph(&gdriver, &gmode, "c:\\tc\\bgi");
        errorcode = graphresult();
        if (errorcode != grOk)  // an error occurred
        {
            cout <<"Graphics error: "<<grapherrormsg(errorcode)
```

```
              <<"Press any key to exit";
          cin.get();
          exit(1); // terminate with an error code
      }
                                  // draw the circle
      setbkcolor(12);
      for(int r=20;r< xradius;r+=10)
      {
          circle(x,y,r);
          outtextxy(250,450,"This is a Circle");
          cin.get();
          cleardevice();
      }
      closegraph();
      return 0;
}
```

لرسم دائرة معينـة معلوم قطرها ينبغي استخدام الدالـة التـي لهـا الشـكل التالي:-

```
void circle(int x,int y,int radius);
```

حيث مركزهـا النقطـة y,x وقطرهـا radius بعـد الاعـلان عـن بعـض المتغيرات وتحديد قيمة قطر الدائرة xradius ومركزها المتمثل في المتغيرين y,x وبعد التأكد من انه تم فتح برنامج الرسم ، تـأتي جملـة for التـي مهمتهـا تكرار عدد من الدوال ومنها دالة رسم الدائرة

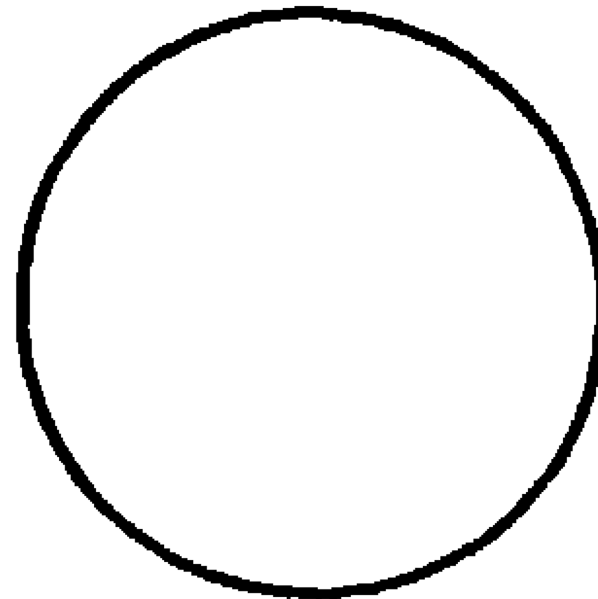
```
circle(x,y,r);
```

ودالة

```
outtextxy(250,450,"This is a Circle");
```

التي عن طريقها يتم طباعة الجملة This is a Circle تحت الدائرة المرسومة وبمجرد الضغط على مفتاح الادخال يتم زيادة قطر الدائرة عـن طريـق المتغير

r وبالتالي تكبيرها، وهكذا حتي تنتهي جملة for ويقفل برنـامج الرسـم، وفيمـا يلي شكل لواحدة من الدوائر الناتجة من تنفيذ هذا البرنامج.



This is a Circle

مثـال 14-4) نقدم الآن برنامجـا آخـر مهمتـه القيـام برسم قطـاعين (Ellipse) الاول فارغ المحتوي والثاني مخطط.

```
#include <graphics.h>
#include <conio.h>
main()
{
      int gdriver = DETECT, gmode, errorcode;
      int stangle = 0, endangle = 360;
      int xradius = 100, yradius = 50;
      int x=getmaxx()/2+350 ,  y=getmaxy()/2+280;
                  // initialize graphics
      initgraph(&gdriver, &gmode, "c:\\tc\\bgi");
                  // draw ellipse
      setbkcolor(12);
      setfillstyle(6,7);
      ellipse(x,y,stangle, endangle,xradius, yradius);
      outtextxy(250,450," This is an ellipse");
      cin.get();
      cleardevice();
                  // draw fillellipse
      setbkcolor(9);
      setfillstyle(6,5);
      fillellipse(x,y,xradius, yradius);
      outtextxy(250,450," This is a fillellipse");
      cin.get();
      cleardevice();
      cin.get();
```

```
        cleardevice();
        closegraph();
        return 0;
}
```

في بداية هذا البرنامج وبعد الاعلان عن بعض المتغيرات جاء استخدام الدالة setfillstyle() التي لها الشكل

setfillstyle(int pattern,int color);

ومهمتها تلوين سطح الشاشة بلون معين ، وحتي يمكن رسم القطاع ellipse أو أي شكل دائرى تستخدم الدالة ellipse التي لها الشكل التالي:-

void ellipse(int x,int y,int stangle, int endangle, int xradius, int yradius);

حيث المتغيران y,x يمثلان مركز الشكل المراد رسمه بينما stangle مهمته تحديد زاوية البداية وendangle مهمته تحديد زاويـة النهاية في حين xradius لتحديد القطر الافقي و yradius لتحديد القطر العمودي.

بعد تحديد زاوية البداية والنهاية وتحديد مركز القطاع واعطاء اللون المناسب لشاشة العرض ثم رسم القطاع الخالي من أي تخطيط بداخلـه ، يلي ذلك كتابة الجملة This is an ellipse تحت هذا القطاع وبمجرد الضغط على مفتاح الادخال يتم تنظيف الشاشة وظهور القطاع المحشو fillellipse من الداخل بخطوط وتم هذا باستخدام الدالة

fillellipse(x,y,xradius, yradius);

التي لها الشكل

void fillellipse(int x,int y,int xradius,int yradius);

مع طباعة الجملة This is a fillellipse تحت هذا القطاع، وفيما يلي الرسم المشابه للقطاع الاول الناتج من تنفيذ هذا البرنامج.



This is an ellipse



This is a fillellipse

مثال 5-14) البرنامج الموالي مهمته القيام برسم ثلاث دوائر متداخلة مع بعضها باستخدام الفصائل.

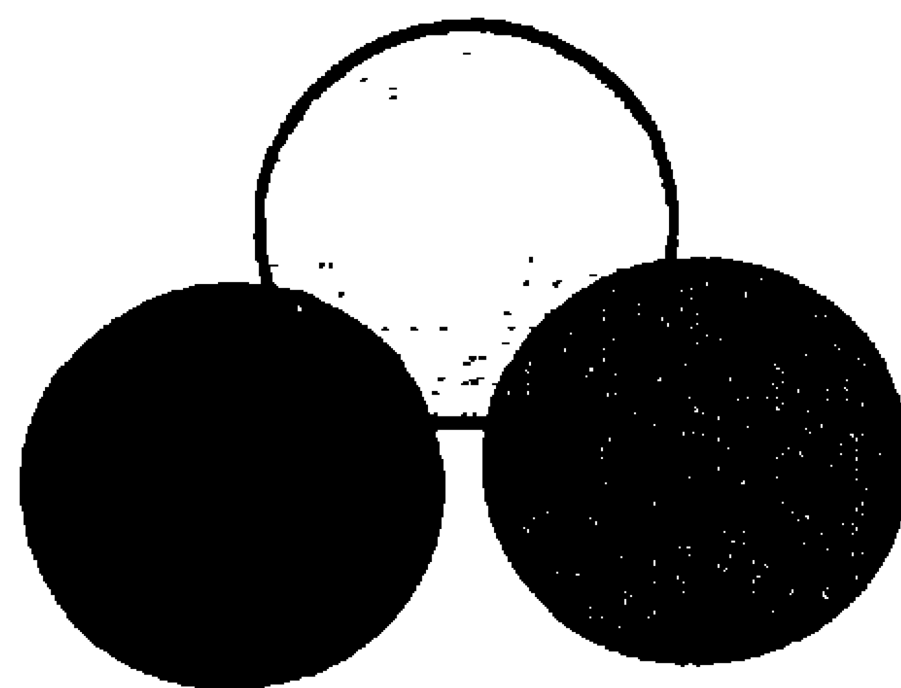
```
#include<graphics.h>
#include<conio.h>
class circle_class
{
	private : int circle_x,circle_y;
		int color1;
		int color2;
		int radius;
	public  : circle_class()
		{ circle_x=0; circle_y=0; color1=0; color2=0; };
		void set(int x,int y,int c1,int c2,int rad)
		{
			circle_x=x;
			circle_y=y;
			color1=c1;
			color2=c2;
			radius=rad;
		};

	void draw()
	{
		setcolor(color1);
		circle(circle_x,circle_y,radius);
		setfillstyle(SOLID_FILL,color2);
```

```
            floodfill(circle_x,circle_y,color1);
      };
};

main()
{
      int gdriver=DETECT, gmode;
      initgraph(&gdriver,&gmode,"c:\\tc\\bgi");
      circle_class b1;
      circle_class b2;
      circle_class b3;
      setbkcolor(12);
      b1.set(200,200,14,4,75);
      b2.set(300,150,14,2,75);
      b3.set(400,200,14,14,75);
      b1.draw();
      b2.draw();
      b3.draw();
      outtextxy(290,300,"Three Circles");
      getch();
      closegraph();
}
```

هنا تم استخدام الدالة circle_class() لاعطاء عدد من المتغيرات الخاصة قيما مبدئية 0 بينما الدالة set() لتخصيص البيانات التي تخص رسم كل دائرة، في حين الدالة draw() مهمتها رسم هذه الدوائر والتي هي بالشكل المشابه للآتي وقت تنفيذ هذا البرنامج.



Three Circles

مثال 14-6) اخراج الحروف والكلمات على هيئة اشكال وفي مواضـع مختلفـة هي مهمة هذا البرنامج.

```
#include <graphics.h>
#include <stdlib.h>
#include <conio.h>
int main(void)
{
        int gdriver = DETECT, gmode,errorcode;
        initgraph(&gdriver, &gmode, "c:\\tc\\bgi");
        errorcode = graphresult();
        if (errorcode != grOk)
        {
              cout <<"Graphics error: "<<endl
                    <<"Press any key to halt:";
              cin.get();
              exit(1);
        }
        setbkcolor(12);
        int size=4;
        settextstyle(DEFAULT_FONT,HORIZ_DIR,size);
        outtextxy(200,270,"C++");
        size=8;
        settextstyle(TRIPLEX_FONT,VERT_DIR,size);
        outtextxy(300,180,"BOOK");
        cin.get();
        closegraph();
        return 0;
}
```

بعد تهيئة النظام الخـاص بالرسم والتـأكد مـن عمليـة نجاحـه ، تـم اعطـاء اللون الاحمر الفاتح للشاشة اعقبها استخدام الدالة settextstyle وشكلها

```
void settextstyle(int font,int direction,int charsize);
```

التي مهمتها التغيير والتحكم في نمط الخط المطلوب عن طريق font والاتجاه أو الطريقة التي يطبع بها الجملة direction والحجم charsize وفي هذا البرنامج تمت كتابة الكلمة C++ باستخدام الخط DEFAULT_FONT بالنقطة (200,270) وبالاتجاه HORIZ_DIR أو القيمة 0 أي من اليمين الى اليسار وبالحجم 4 ، تلا ذلك وفي نفس الشاشة اضهار الكلمة BOOK ولكن بخط TRIPLEX_FONT وبالاتجاه VERT_DIR أو القيمة 1 أي من اسفل الى اعلى، عموما عند تنفيذ هذا البرنامج سينتج عنه ما يلي :-

C++

BOOK

مثال 14-7) نوع آخر من دوال الرسم بهذه اللغة يقدمه البرنامج الموالي

```
#include <graphics.h>
#include <conio.h>
#include <dos.h>
void main()
{
        int font_size=4;
        int driver =DETECT,mode;
        int x=0,y=0,X,Y;
        char *string ="HELLO USER";
        unsigned int size;
        initgraph(&driver,&mode,"c:\\tc\\bgi");
        while(!kbhit())
        {
```

```
            sound(440);
            setbkcolor(1);
            settextstyle(DEFAULT_FONT,HORIZ_DIR,font_size);
            outtextxy(x,y,string);
            delay(80);
            cleardevice();
            x+=24;
            if(x>getmaxx())
            {
                 y+=50;
                 y=(y<getmaxy()-100) ? y : 0;
                 x=0;
            }
            nosound();
       }
       closegraph();
}
```

عند تنفيذ هذا البرنامج ينتج عنه ظهور وتحريك الجملة HELLO USER على الشاشة باللون الابيض بالمستوى الافقي أي من اليسار الى اليمين مع الصوت المتقطع باعلى الشاشة ثم السطر الموالي وهكذا يتكرر ضهور هذه الجملة حتي الضغط على مفتاح الادخال Enter لإنهاء البرنامج.

في هذا البرنامج تم استخدام بعض الدوال التي سبق شرحها مثل settextstyle لاعطاء مواصفات الكتابة من حيث الشكل والاتجاه والحجم بالاضافة الى الدالة outtextxy التي مهمتها طباعة الجملة في المكان المحدد بالنقطة y,x التي تتغير من خلال جملة if والمؤثر الشرطي داخل جملة while وهذا يحدث عندما تصل الجملة المطبوعة في اقصي يمين الشاشة أي عندما يتحقق الشرط x>getmaxx() حيث تصبح قيمة x اكبر من قيمة احداثيات المحور السيني، عندها تضاف القيمة 50 الى قيمة المتغير y وتقارن باستخدام المؤثر الشرطي مع احداثيات المحور الصادي وفي نفس الوقت يعطي المتغير x القيمة 0 بالتالي يعاد ظهور الجملة مرة أخرى على يمين الشاشة، وفي اثناء

التحرك يحدث صوت مصاحب لنتيجة استخدام الدالة sound التي تعطي ترددا frequency معينا على اجهزة الحاسب الشخصي مع ايقاف تنفيذ البرنامج لمـدة معينــة وذلــك بتحديـد تلـك المـدة وتكـون بـالجزء مـن ألـف مـن الثانيـة (milliseconds) باستخدام الدالة delay() ليحدث تقطعات في خروج الصوت.

مثال 14-8) البرنامج التالي يبين طريقة رسم صندوق ذى ثلاثة ابعاد three-dimensional bar باستخدام بعض الدوال الجاهزة.

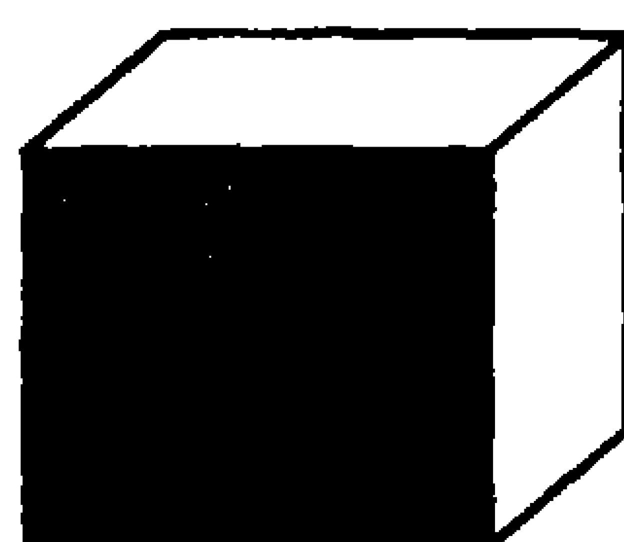
```
#include <graphics.h>
#include <conio.h>
main()
{
        int gdriver = DETECT, gmode, errorcode;
        int left, top, right, bottom;
                // initialize graphics
        initgraph(&gdriver, &gmode, "c:\\tc\\bgi");
        left = getmaxx() / 2 - 100;
        top = getmaxy() / 2 - 100;
        right = getmaxx() /2 + 100;
        bottom = getmaxy() /2 + 100;
        setbkcolor(12);
                // draw a three dimensional rectangular bar
        bar3d(left,top,right,bottom,(right-left)/4,1);
        outtextxy(250,450," This is a bar3d rectangular");
        getch();
        cleardevice();
        closegraph();
        return 0;
}
```

في هذا البرنامج تم استخدام الدالة bar3d التـي مهمتهـا رسـم صنـدوق ذى ثلاثة ابعاد وهي تأخذ الشكل التالي:-

```
void bar3d(int left,int top,int right,int bottom,int depth,int topflag);
```

حيث top,left الزاوية العليا من جهة يسار الشاشة و bottom,right الزاوية السفلي من جهة يمين الشاشة في حين depth العمق حيث إستخدمت المعادلة (right-left)/4 في هذا البرنامج لتمثل نسبة 25 في المائة من عرض الشكل المرسوم بينما topflag يمثل الغطاء العلوي من الصندوق.

هذا البرنامج ينتج عنه الشكل التالي وقت تنفيذه



This is a bar3d rectangular

مثال 9-14) نقدم الآن برنامجاً مهمته رسم خط مستقيم يمثل المحور السيني (x axis) باستخدام الدالة المشهورة في هذا الخصوص.

```
#include <graphics.h>
#include <conio.h>
int main(void)
{
	int gdriver = DETECT, gmode, errorcode;
	initgraph(&gdriver, &gmode, "c:\\tc\\bgi");
	setbkcolor(2);
	line(100,250,500,250);
	setcolor(1);
	outtextxy(105,260,"1");
	outtextxy(150,260,"2");
	outtextxy(200,260,"3");
	outtextxy(250,260,"4");
	outtextxy(300,260,"5");
	outtextxy(350,260,"6");
	outtextxy(400,260,"7");
	outtextxy(450,260,"8");
	outtextxy(250,330,"This is X axis line");
```

```
        getch();
        cleardevice();
        closegraph();
        return 0;
}
```

في هذا البرنامج استخدمت الدالة line والتي تأخذ الشكل التالي :-

```
line(int x1,int y1,int x2,int y2);
```

ومهمتها رسم خط مستقيم بداية من النقطة (x1,y1) وحتـى النقطـة (x2,y2) وعليه تم رسم الخط الافقي المستقيم أي المحـور السـيني (x axis) بدايـة مـن النقطـة (100,250) التـي تمثـل (x1,y1) وحتـي النقطـة (500,250) التـي تمثـل (x2,y2) أيضا طباعة الارقـام مـن 1 الـى 8 تحـت هـذا الخـط باسـتخدام الدالـة outtextxy واخيرا اظهار الرسالة This is X axis line تحتـه ، وفيمـا يلـي نـاتج تنفيذ هذا البرنامج:-

```
______________________________
  1  2  3  4  5  6  7  8
      This is X axis line
```

مثال 14-10) يمكن اعادة كتابـة البرنـامج بالمثـال السـابق(14-9) مـع اسـتبدال الدالة line والدالة outtextxy لتأخذ الشكل التالي :-

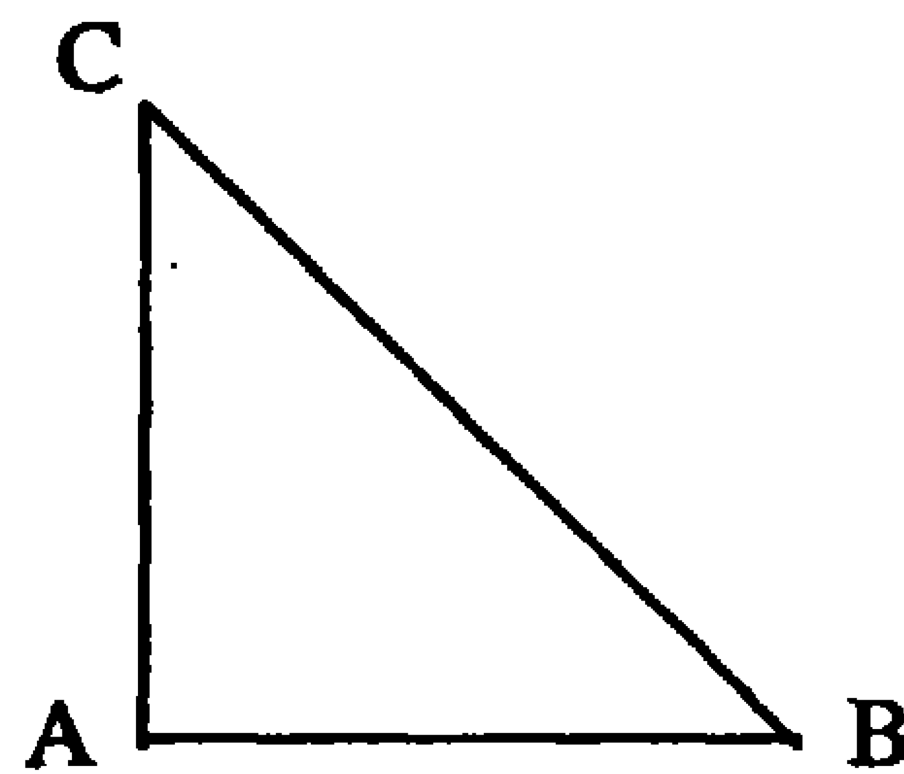
```
line(100,300,100,50);
outtextxy(80,50,"50");
outtextxy(80,100,"40");
outtextxy(80,150,"30");
outtextxy(80,200,"20");
outtextxy(80,250,"10");
outtextxy(80,300,"0");
outtextxy(100,330,"This is Y axis line");
```

وبالتالي الحصول على خط مستقيم الذي يمثل المحور الصادي (y axis) بداية من النقطة (100,300) وحتى النقطة (100,50) مع اظهار بعض الارقام بجانب هذا الخط.

مثال 14-11): يمكن الحصول على رسم مثلث باستخدام الدالة line والدالة outtextxy عن طريق البرنامج التالي:-

```
#include <graphics.h>
#include <conio.h>
int main(void)
{
	int gdriver = DETECT, gmode, errorcode;
	initgraph(&gdriver, &gmode, " c:\\tc\\bgi ");
	setbkcolor(7);
	line(100,300,100,50); //  y axis from (100,300) to (100,50)
	line(100,50,300,300); // x axis from (100,50) to (300,300)
	line(100,300,300,300);// from (100,300) to (300,300)
	setcolor(1);
	outtextxy(80,300,"A");  .
	outtextxy(330,300,"B");
	outtextxy(80,50,"C");
	outtextxy(150,330,"This is a triangle");
	getch();
	cleardevice();
	closegraph();
	return 0;
}
```

البرنامج عند تنفيذه سوف ينتج عنه رسم المثلث بالشكل التالي :-



This is a triangle

مثـال 14-12) فيمـا يلـي برنـامجـا كـاملا يتـم فيـه قـراءة الكميـات المباعـة مـن البضائع في الاربعة الاشـهر الاولـي مـن السـنة فـي اثنين مـن المخـازن مـع توضيح العلاقة بواسطة الرسم بين اكبر كميـة مباعـة فـي المخزنيـن والشـهر المقابل لهذه الكمية.

```
#include <iostream.h>
#include <string.h>
#include <graphics.h>
#include <stdlib.h>
#include <conio.h>
int main(void)
{
        int gdriver = DETECT, gmode, errorcode;
        int month1[10],month2[10],month3[10],month4[10];
        int max1,max2,max3,max4;
        char string[10],string1[10],string2[10],string3[10],string4[10];
        char str[10];
        cout<<"Enter sale year ==>:";
        cin>>str;
        for(int i=1;i<=2;i++)
        {
             cout<<"Enter sale quantity in 4 month for store "<<i<<" ==>: ";
             cin>>month1[i]>>month2[i]>>month3[i]>>month4[i];
        }
        max1=month1[1];max2=month2[1];max3=month3[1];max4=month4[1];
        for(i=1;i<=2;i++)
        {
             if(max1<month1[i])
```

```
        max1=month1[i];
        if(max2<month2[i])
        max2=month2[i];
        if(max3<month3[i])
        max3=month3[i];
        if(max4<month4[i])
        max4=month4[i];
    }
    initgraph(&gdriver, &gmode, "c:\\tc\\bgi");
    setbkcolor(12);
    line(100,250,500,250);
    line(100,400,100,50);
    line(100,250-max1*3,100,250-max1*3);
    line(100,250-max1*3,130,250-max2*3);
    line(130,250-max2*3,160,250-max3*3);
    line(160,250-max3*3,190,250-max4*3);
    setcolor(1);
    outtextxy(450,260,"MONTHS");
    outtextxy(100,40,"QUANTITY");
    outtextxy(60,250,"0");
    outtextxy(60,220,"10");
    outtextxy(60,190,"20");
    outtextxy(60,160,"30");
    outtextxy(60,130,"40");
    outtextxy(60,100,"50");
    outtextxy(105,260,"1");
    outtextxy(130,260,"2");
    outtextxy(160,260,"3");
    outtextxy(190,260,"4");
    setcolor(15  );
    settextstyle(7,HORIZ_DIR,3);
    outtextxy(5,7,"SALES REPORT YEAR ");
    setcolor(15);
    settextstyle(7,HORIZ_DIR,4);
    outtextxy(300,5,str);
    getch();
    cleardevice();
    closegraph();
    return 0;
}
```

بعد الاعلان عن بعض المتغيرات المناسبة وتهيئة نظام الرسم للعمل، جاءت المطالبة بادخال السنة التي تم فيها البيع ولتكن سنة 1999 مثلا

```
Enter sale year ==>:1999
```

تلا ذلك المطالبة بادخال الكميات المباعة من المخزن الاول وهي

```
Enter sale quantity in 4 month for store 1 ==>:25  0  30  10
```

ثم كمية المبيعات في المخزن الثاني وهي

```
Enter sale quantity in 4 month for store 2 ==>:30  0  50  20
```

اعقب ذلك المقارنة بين عدد الكميات المباعة في كل شهر بالمخزنين لتحديد القيمة الكبري ثم جرت تهيئة نظام الرسم للعمل واعطاء اللون الخلفي المناسب للشاشة أعقبها استخدام الدالة line وعليه تم رسم الخط الافقي أي المحور السيني (x axis) الذي يمثل الاشهر عن طريق الامر:

```
line(100,250,500,250);
```

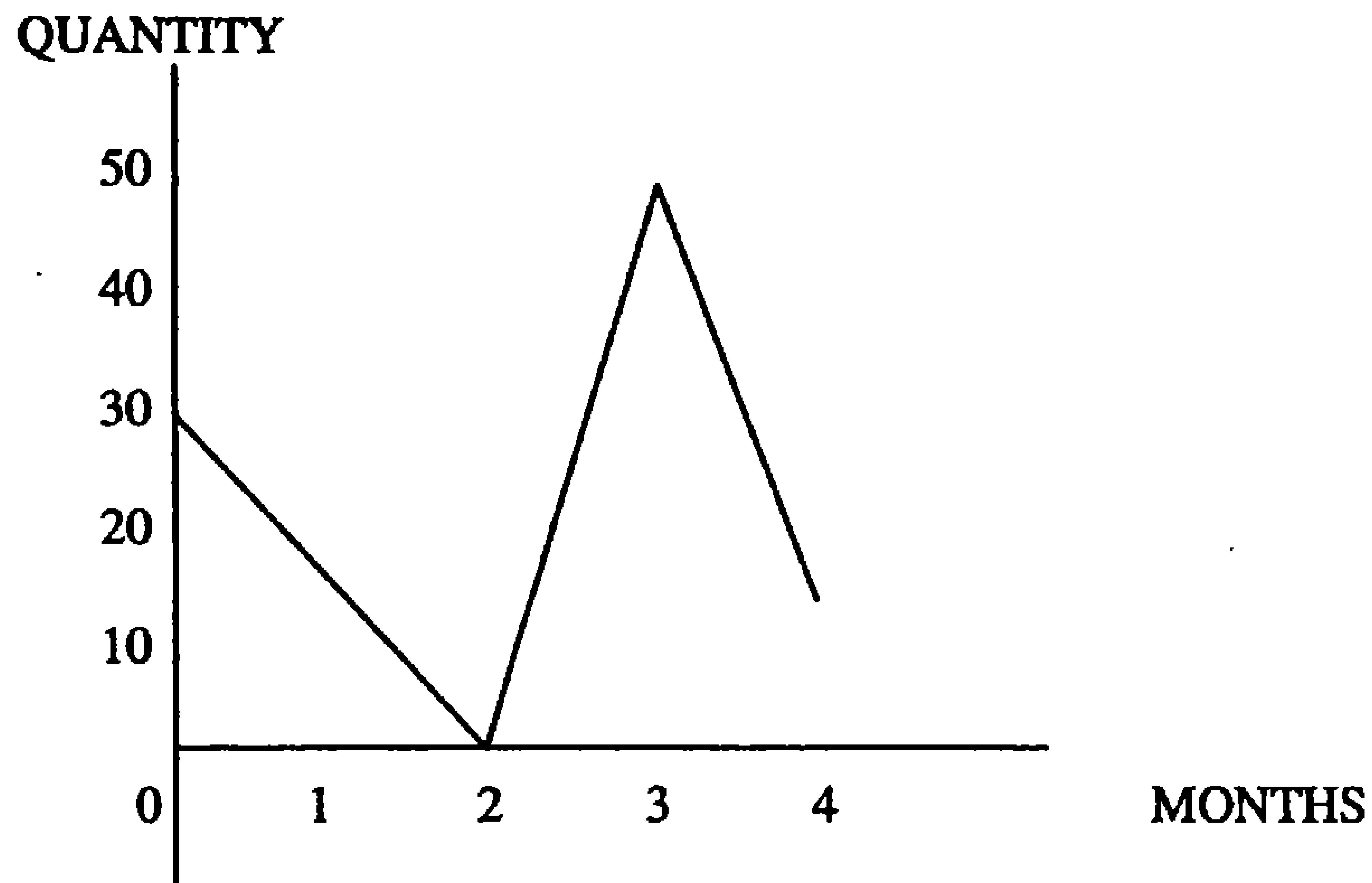
بينما الامر:

```
line(100,400,100,50);
```

لرسم الخط العمودي أي المحور الصادي (y axis) الذي يمثل الكميات المباعة في كل شهر.

وبنفس الطريقة تم رسم بقية الخطوط بين اعلى كمية مبيعات والشهر الذي وقعت فيه باستخدام نفس الدالة line مع استخدام الدالة outtextxy لتحديد النقاط التي تمثلها الاشهر على المحور السيني والنقاط التي تمثلها الكميات المباعة على المحور الصادي واخيرا بعض الدوال الاخرى التي تم شرحها سابقا .

وهذا هو ناتج البرنامج وقت تنفيذه



مثال 13-14) البرنامج مهمته استقبال عدد الطلبة المنسبين الى قسم الحاسب الآلي خلال ثلاث سنوات متتالية مع رسم مدرج تكراري يبين نسبة الطلبة في كل سنة.

```
#include<conio.h>
#include<dos.h>
#include<stdlib.h>
#include<graphics.h>
#include<iostream.h>
#define tc textcolor(1)
#define cls clrscr()

int main(void)
{
        cls;
                // request autodetection
        int gdriver=DETECT,errorcode,gmode;
        float x[3],xx[3];
                // initialize graphics moad
        initgraph(&gdriver,&gmode, "c:\\tc\\bgi");
                // read result of initialization
```

```
errorcode=graphresult();
if(errorcode !=grOk )   // an erroe occurred
{
     cout <<"graphics erroe"<<grapherrormsg(errorcode)
          <<"press any key to halt:";
     cin.get();
     exit(1);
}
tc;
char *years[3]={"1997","1998","1999"};
for (int i=1;i<=3;i++)
{
     cout <<"Enter number of student in year "<<years[i-1]
          <<" please ==>:";
     cin>>x[i];
}
cls;
x[1]=x[1]/10;
x[2]=x[2]/10;
x[3]=x[3]/10;
line(100,350,550,350);
line(150,400,150,50);
xx[1]=350-x[1]*3;
xx[2]=350-x[2]*3;
xx[3]=350-x[3]*3;
outtextxy(550,360,"year");
outtextxy(100,30,"student number");
outtextxy(110,320,"100");
outtextxy(110,290,"200");
outtextxy(110,260,"300");
outtextxy(110,230,"400");
outtextxy(110,200,"500");
outtextxy(110,170,"600");
outtextxy(110,140,"700");
outtextxy(110,110,"800");
outtextxy(110,80,"900");
outtextxy(110,50,"1000");
int  postion=60;
int step_x=70;
int current_x=step_x*3;
```

```
    int s=(600-100)/9;
    for(i=1;i<=3;i++)
    {
        setfillstyle(4+(i*1),10*(i+1));
        bar(100+100*i,xx[i]+i*1,current_x+step_x/2,350);
        outtextxy(160+postion,360,years[i-1]);
        sound(110+s*i);delay(200);
        current_x+=step_x+step_x/2;
        postion+=95;
    }
    nosound();
    getch();
    closegraph();
    return 0;
}
```

بعد تنفيذ هذا البرنامج الذي استخدمت فيه عدد من الدوال الجاهزة ، وبعد استقباله لعدد الطلبة المنسبين للقسم في السنة الاولي كالتالي:-

Enter number of student in year 1997 please ==>:400

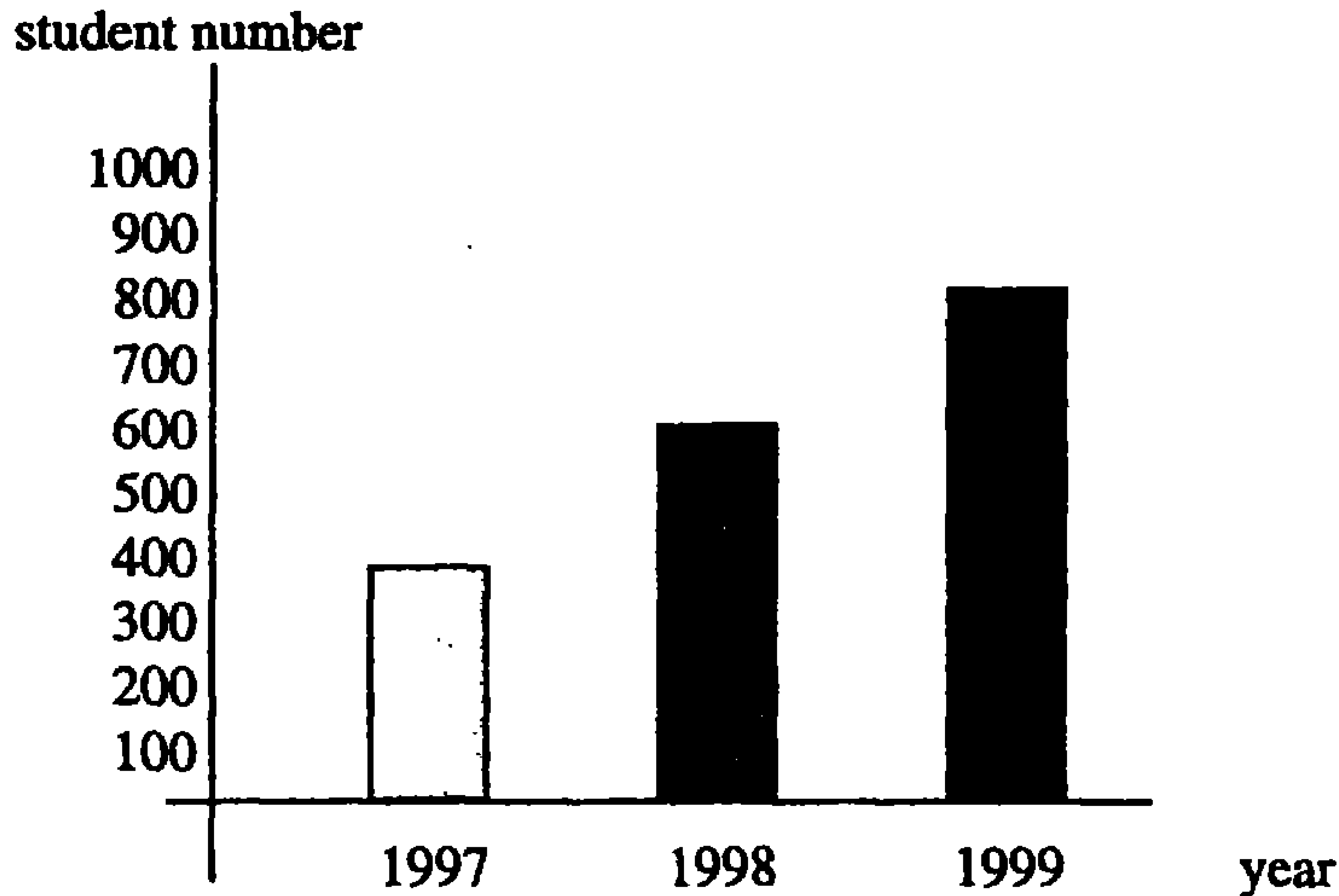
وفي السنة الثانية

Enter number of student in year 1998 please ==>:600

واخيرا السنة الثالثة

Enter number of student in year 1999 please ==>:800

ينتج عنه الشكل الموضح التالي :-



مثال 14-14) لمساعدة المتعلم واعطائه القدرة على استعمال هذه اللغة في أعمال الاحصائيات والاستبيانات وتمكينه من استخدام الدوال بلغة سي++ وتطويعها للعديد من الرسومات ، عليه فاننا نقدم هذا البرنامج الذي مهمته استقبال عدد من الطلبة المنتسبين الى اربعة اقسام بالكلية ومن ثم توضيح العلاقة بين هذه الاقسام من حيث عدد الطلبة المنسبين اليها عن طريق دائرة مكونة من عدد من القطاعات حيث كل قطاع يمثل قسما من اقسام الكلية الاربعة.

```
#include <graphics.h>
#include <stdlib.h>
#include <iostream.h>
#include <conio.h>
main()
{
	float degrees1,degrees2,degrees3,degrees4;
	int gdriver = DETECT, gmode, errorcode;
	char string[15];
	int dep[4],angl1,angl2,angl3,angl4,i,sum=0;
```

```
clrscr();
cout<<"Type number of student in computer please ==>" ;
cin>>dep[1];
cout<<"Type number of student in statistics please ==>" ;
cin>>dep[2];
cout<<"Type number of student in physics please ==>" ;
cin>>dep[3];
cout<<"Type number of student in mathematics ==>" ;
cin>>dep[4];
sum=sum+dep[1]+dep[2]+dep[3]+dep[4];
degrees1=(float)dep[1] /sum;
degrees2=(float)dep[2] /sum;
degrees3=(float)dep[3] /sum;
degrees4=(float)dep[4] /sum;
angl1=degrees1*360;
angl2=degrees2*360;
angl3=degrees3*360;
angl4=degrees4*360;
initgraph(&gdriver, &gmode, "C:\\TC\\BGI\\");
                        // read result of initialization
errorcode = graphresult();
if (errorcode != grOk)  // an error occurred
{
     cout <<"Graphics error: "<<grapherrormsg(errorcode)
          <<"Press any key to halt:";
     cin.get();
     exit(1); // terminate with an error code
}
int xx=getmaxx()-30,yy=15;
setbkcolor(12);
itoa(degrees1*100,string,10);
outtextxy(190,110,string);
outtextxy(210,110,"%");
itoa(degrees2*100,string,10);
outtextxy(190,160,string);
outtextxy(210,160,"%");
itoa(degrees3*100,string,10);
outtextxy(190,210,string);
outtextxy(210,210,"%");
itoa(degrees4*100,string,10);
outtextxy(190,260,string);
```

```
        outtextxy(210,260,"%");
        setfillstyle(6,1);
        setfillstyle(6,8);
        int size=4;
        settextstyle(TRIPLEX_FONT,HORIZ_DIR,size);
        outtextxy(150,400,"DATA DESCRIPTION");
        setfillstyle(6,4);
        pieslice(420,240,0,angl1,100);
        setcolor(15);
        size=2;
        settextstyle(TRIPLEX_FONT, HORIZ_DIR,size);
        outtextxy(50,10,"THE PERCENTS OF DEPARTMENTS IN FACULTY");
        settextstyle(1, HORIZ_DIR,2);
        outtextxy(20,100,"IN COMPUTER");
        setfillstyle(6,15);
        pieslice(420,240,angl1,angl1+angl2,100);
        setcolor(15);
        settextstyle(1, HORIZ_DIR,size);
        outtextxy(20,150,"IN STATISTICS");
        setfillstyle(6,3);
        pieslice(420,240,angl1+angl2,angl1+angl2+angl3,100);
        setcolor(15);
        settextstyle(1, HORIZ_DIR,size);
        outtextxy(20,200,"IN PHYSICS");
        setfillstyle(6,1);
        pieslice(420,240,angl1+angl2+angl3,360,100);
        setcolor(15);
        settextstyle(1, HORIZ_DIR,size);
        outtextxy(20,250,"IN MATHEMATICS");
        setfillstyle(1,4);
        bar(0,100,10,120);
        setfillstyle(1,15);
        bar(0,150,10,170);
        setfillstyle(1,3);
        bar(0,200,10,220);
        setfillstyle(1,1);
        bar(0,250,10,270);
        getch();
        closegraph();
        return 0;
}
```

في بداية هذا البرنامج تم الاعلان عن بعض المتغيرات ثم جرى ادخال عدد الطلبة المنسبين لكل قسم من اقسام الكلية الاربعة تلا ذلك ايجاد مجموع الطلبة بالكلية وتخزينه في المتغير sum مع ايجاد نسبة الطلبة في القسم الاول وحساب الزاوية التي بواسطتها ستمثل القطاعات الدائرية حسب المعادلة degree1

الزاوية = النسبة مضروبة في 360

بعد التأكد من فتح نظام الرسم جاء استخدام الدالة itoa التي مهمتها تحويل القيمة الصحيحة المخزنة في المتغير degrees1 لسلسلة حرفية وتخزينها في المتغير الحرفي string حتى يمكن استخدام الدالة ()outtextxy لطباعة هذه السلسلة في المكان المناسب ، ايضا تم اعطاء القطاعات الدائرية اللون والشكل المناسبين عن طريق الدالة ()setfillstyle ، اما الدالة pieslice فمهمتها رسم القطاعات الدائرية وشكلها

```
void pieslice(int x,int y,int stangle,int endangle,int radius);
```

حيث y,x نقطة تحديد المركز و endangle,stangle بداية ونهاية زاوية القطاع اما radius فيعني قطر القطاع ، أخيرا استدعيت الدالة bar لرسم مربعات صغيرة تبين شكل القطاع المرسوم.

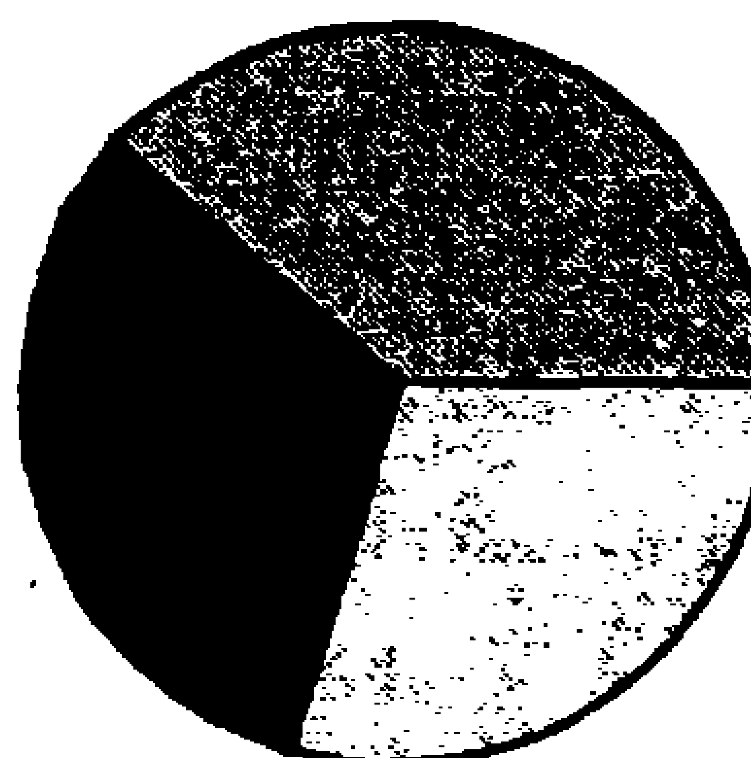
عند تنفيذ هذا البرنامج وادخال اعداد الطلبة لكل قسم كما يلي :-

```
Type number of student in computer please ==>40
Type number of student in statistics please ==>20
Type number of student in physics please ==>10
Type number of student in mathematics ==>30
```

ينتج عنه ظهور شاشة العرض بالشكل الموالي

THE PERCENTS OF DEPARTMENTS IN FACULTY

IN COMPUTER	40%
IN STATISTICS	20%
IN PHYSICS	10%
IN MATHEMATICS	30%



DATA DECRIPTION

الملاحـــق

ملحق (1)
جدول أولويات تنفيذ العمليات

المؤثر	الأولوية
() , [] , -> , :: , .	من اليسار الى اليمين
! , ~ , + , -(unary) , ++ , -- , & , * , sizeof(type)	من اليمين الى اليسار
* , / , %	من اليسار الى اليمين
+ , -	من اليسار الى اليمين
<< , >>	من اليسار الى اليمين
< , <= , > , >=	من اليسار الى اليمين
== , !=	من اليسار الى اليمين
&	من اليسار الى اليمين
^	من اليسار الى اليمين
|	من اليسار الى اليمين
&&	من اليسار الى اليمين
||	من اليسار الى اليمين
?:	من اليمين الى اليسار
= , *= , /= , %= , += , -= , &= , ^= , |= , <<= , >>=	من اليمين الى اليسار
(comma operator)	من اليسار الى اليمين

ملحق (2)

جدول نظام الشفرة الامريكية المعيارية لتبادل المعلومات (ASCII)

القيمـــــة بالنظـــــام									
عشري	ثنائي	ثماني	ستة عشري	ascii	عشري	ثنائي	ثماني	ستة عشري	ascii
0	00000000	000	0x0	NUL	65	01000001	101	0x41	A
1	00000001	001	0x1	SOH	66	01000010	102	0x42	B
2	00000010	002	0x2	STX	67	01000011	103	0x43	C
3	00000011	003	0x3	ETX	68	01000100	104	0x44	D
4	00000100	004	0x4	EOT	69	01000101	105	0x45	E
5	00000101	005	0x5	ENQ	70	01000110	106	0x46	F
6	00000110	006	0x6	ACK	71	01000111	107	0x47	G
7	00000111	007	0x7	BEL	72	01001000	110	0x48	H
8	00001000	010	0x8	BS	73	01001001	111	0x49	I
9	00001001	011	0x9	HT	74	01001010	112	0x4A	J
10	00001010	012	0xA	LF	75	01001011	113	0x4B	K
11	00001011	013	0xB	VT	76	01001100	114	0x4C	L
12	00001100	014	0xC	FF	77	01001101	115	0x4D	M
13	00001101	015	0xD	CR	78	01001110	116	0x4E	N
14	00001110	016	0xE	SO	79	01001111	117	0x4F	O
15	00001111	017	0xF	SI	80	01010000	120	0x50	P
16	00010000	020	0x10	DLE	81	01010001	121	0x51	Q
17	00010001	021	0x11	DC1	82	01010010	122	0x52	R
18	00010010	022	0x12	DC2	83	01010011	123	0x53	S
19	00010011	023	0x13	DC3	84	01010100	124	0x54	T
20	00010100	024	0x14	DC4	85	01010101	125	0x55	U

القيمــــــة بالنظــــــام									
عشري	ثنائي	ثماني	ستة عشري	ascii	عشري	ثنائي	ثماني	ستة عشري	ascii
21	00010101	025	0x15	NAK	86	01010110	126	0x56	V
22	00010110	026	0x16	SYN	87	01010111	127	0x57	W
23	00010111	027	0x17	ETB	88	01011000	130	0x58	X
24	00011000	030	0x18	CAN	89	01011001	131	0x59	Y
25	00011001	031	0x19	EM	90	01011010	132	0x5A	Z
26	00011010	032	0x1A	SUB	91	01011011	133	0x5B	[
27	00011011	033	0x1B	ESC	92	01011100	134	0x5C	\
28	00011100	034	0x1C	FS	93	01011101	135	0x5D	]
29	00011101	035	0x1D	GS	94	01011110	136	0x5E	^
30	00011110	036	0x1E	RS	95	01011111	137	0x5F	_
31	00011111	037	0x1F	US	96	01100000	140	0x60	`
32	00100000	040	0x20	SP	97	01100001	141	0x61	a
33	00100001	041	0x21	!	98	01100010	142	0x62	b
34	00100010	042	0x22	"	99	01100011	143	0x63	c
35	00100011	043	0x23	#	100	01100100	144	0x64	d
36	00100100	044	0x24	$	101	01100101	145	0x65	e
37	00100101	045	0x25	%	102	01100110	146	0x66	f
38	00100110	046	0x26	&	103	01100111	147	0x67	g
39	00100111	047	0x27	'	104	01101000	150	0x68	h
40	00101000	050	0x28	(	105	01101001	151	0x69	I
41	00101001	051	0x29	)	106	01101010	152	0x6A	j
42	00101010	052	0x2A	*	107	01101011	153	0x6B	k
43	00101011	053	0x2B	+	108	01101100	154	0x6C	l

القيمـــــــة بالنظـــــــام									
ascii	ستة عشري	ثماني	ثنائي	عشري	ascii	ستة عشري	ثماني	ثنائي	عشري
m	0x6D	155	01101101	109	,	0x2C	054	00101100	44
n	0x6E	156	01101110	110	-	0x2D	055	00101101	45
o	0x6F	157	01101111	111	.	0x2E	056	00101110	46
p	0x70	160	01110000	112	/	0x2F	057	00101111	47
q	0x71	161	01110001	113	0	0x30	060	00110000	48
r	0x72	162	01110010	114	1	0x31	061	00110001	49
s	0x73	163	01110011	115	2	0x32	062	00110010	50
t	0x74	164	01110100	116	3	0x33	063	00110011	51
u	0x75	165	01110101	117	4	0x34	064	00110100	52
v	0x76	166	01110110	118	5	0x35	065	00110101	53
w	0x77	167	01110111	119	6	0x36	066	00110110	54
x	0x78	170	01111000	120	7	0x37	067	00110111	55
y	0x79	171	01111001	121	8	0x38	070	00111000	56
z	0x7A	172	01111010	122	9	0x39	071	00111001	57
{	0x7B	173	01111011	123	:	0x3A	072	00111010	58
		0x7C	174	01111100	124	;	0x3B	073	00111011	59
}	0x7D	175	01111101	125	<	0x3C	074	00111100	60
~	0x7E	176	01111110	126	=	0x3D	075	00111101	61
DEL	0x7F	177	01111111	127	>	0x3E	076	00111110	62
					?	0x3F	077	00111111	63
					@	0x40	100	01000000	64

ملحق (3)

جدول الوان ++Turbo C

اللون			اللون		
بالارقام	بالحروف	المعنى	بالارقام	بالحروف	المعنى
0	BLACK	اسود	8	DARK GRAY	رمادي غامق
1	BLUE	ازرق	9	LIGHT BLUE	ازرق فاتح
2	GREEN	اخضر	10	LIGHT GREEN	اخضر فاتح
3	CYAN	كحلي	11	LIGHT CYAN	كحلي فاتح
4	RED	احمر	12	LIGHT RED	احمر فاتح
5	MAGENTA	ارجواني	13	LIGHT MAGENTA	ارجواني فاتح
6	BROWN	بني	14	YELLOW	اصفر
7	LIGHT GRAY	رمادي داكن	15	WHITE	ابيض

ملاحظة : يمكن استخدام اللون المطلوب اما بالارقام أو بالحروف

ملحق (4)

مراجع الكتاب

1) Turbo C++ An Introduction to Computing Joel Adams and Sanford Leestma and Larry Nyhoff
Pretice Hall, Inc. New Jersey (1996).

2) An Introduction to Object-Oriented Design in C++
Jo Ellen Perry and Harold D. Levin
Addison-Wesley Company, Inc. (1996)

هذا الكتاب

يتناول واحدة من أهم وأكثر لغات الحاسب الآلي الرائدة في مجال البرمجة في عالمنا المعاصر، وهي لغة C++ حيت تم تقديم شروحات لعدد كبير من الأوامر والبرامج المتنوعة محلولة علميا على جهاز الحاسب الشخصي مع مجموعة من التمرينات في آخر كل فصل، مما يساعد علي إثراء مادته ويجعل كتاباً منهجياً لطلبنا في الجمعات والمعاهد ومساعدة من يرغب في تعلم اللغة بنفسه، حتي يمكنه التعامل مع جهاز الحاسب الآلي الذي أصبح من ضروريات حياتنا اليومية.

الكتب الصادرة للمؤلف

